Exhibit 4

US006799295B2

(12) **United States Patent**
Nguyen

(10) **Patent No.: US 6,799,295 B2**
(45) **Date of Patent: Sep. 28, 2004**

(54) HIGH SPEED TURBO CODES DECODER FOR 3G USING PIPELINED SISO LOG-MAP DECODERS ARCHITECTURE

(75) Inventor: **Quang Nguyen**, Allentown, PA (US)

(73) Assignee: **Icomm Technologies, Inc.**, Wilmington, DE (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 30 days.

(21) Appl. No.: **10/248,245**

(22) Filed: **Dec. 30, 2002**

(65) **Prior Publication Data**

US 2003/0084398 A1 May 1, 2003

Related U.S. Application Data

(63) Continuation-in-part of application No. 10/065,408, filed on Oct. 15, 2002, which is a continuation-in-part of application No. 09/681,093, filed on Jan. 2, 2001.

(51) **Int. Cl.**^{7} **H03M 13/00**
(52) **U.S. Cl.** **714/794**; 714/755
(58) **Field of Search** 714/786, 755, 714/795, 794; 375/265

(56) **References Cited**

U.S. PATENT DOCUMENTS

6,304,995	B1	*	10/2001	Smith et al.	714/786
6,484,283	B2	*	11/2002	Stephen et al.	714/786
6,510,536	B1	*	1/2003	Crozier et al.	714/755

OTHER PUBLICATIONS

Abeta et al., Performance of very low rate channel coding in W–CDMA reverse link, 2002, IEEE, p. 1–5.*
Fanucci et al., VLSI design of a high speed Turbo Decoder for 3rd generation satellite communication, 2002, IEEE, p. 509–512.*
Xu et al., Implementation of a new structure of Turbo decoder with 384Kbps in 3G, Feb. 5, 2001, ASIC & system state key Lab, p. 1–7.*

Dr. Woodard, Implementation of high rate Turbo decoders for third generation mobile communications, 1999, Institute of Electrical Engineers, p. 12/1–12/6.*

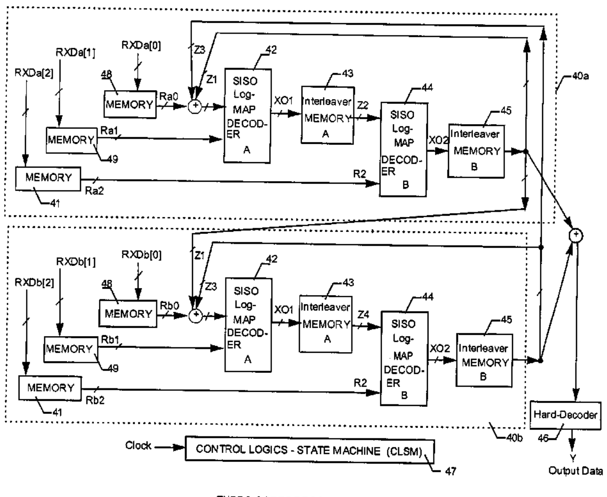
* cited by examiner

Primary Examiner—Albert Decady
Assistant Examiner—Shelly A Chase
(74) *Attorney, Agent, or Firm*—Mintz Levin Cohn Ferris Glovsky and Popeo PC

(57) **ABSTRACT**

A baseband processor is provided having Turbo Codes Decoders with Diversity processing for computing signals from separate antennas. The invention decodes multipath signals that have arrived at the terminal via different routes after being reflected from buildings, trees or hills. The Turbo Codes Decoder with Diversity processing increases the signal to noise ratio (SNR) more than 6 dB which enables the 3rd Generation Wireless system to deliver data rates from up to 2 Mbit/s. The invention provides several improved Turbo Codes Decoder methods and devices that provide a more suitable, practical and simpler method for implementation a Turbo Codes Decoder in ASIC or DSP codes. A plurality of parallel Turbo Codes Decoder blocks are provided to compute soft-decoded data RXDa, RXDb from two different receiver path. Several pipelined Log-MAP decoders are used for iterative decoding of received data. A Sliding Window of Block N data is used on the inputted data for pipeline operations. In a pipeline mode, a first decoder A decodes block N data from a first source, while a second decoder B decodes block N data from a second source during the same clock cycle. Pipelined Log-MAP decoders provide high speed data throughput and one output per clock cycle.

31 Claims, 22 Drawing Sheets



TURBO CODES DECODER SYSTEM BLOCK DIAGRAM

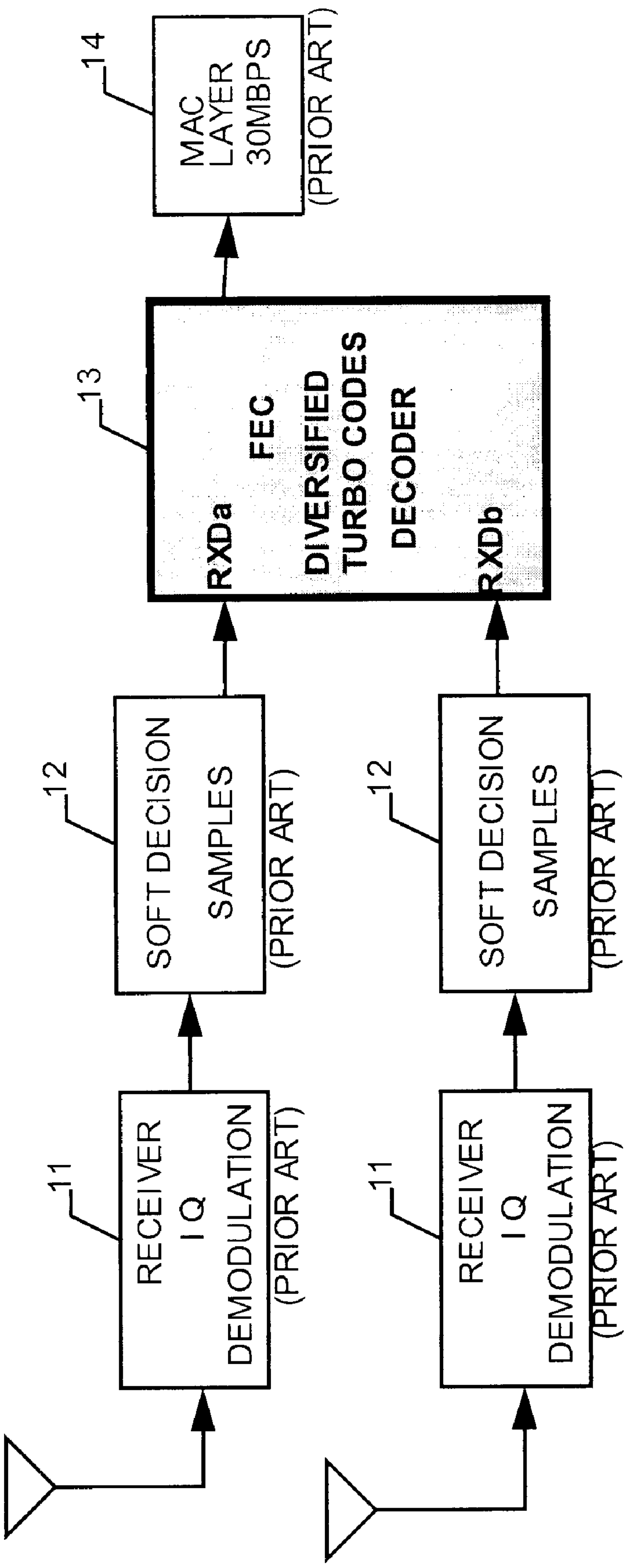


FIGURE 1. TYPICAL DIVERSIFIED RECEIVER FUNCTIONAL BLOCK DIAGRAM

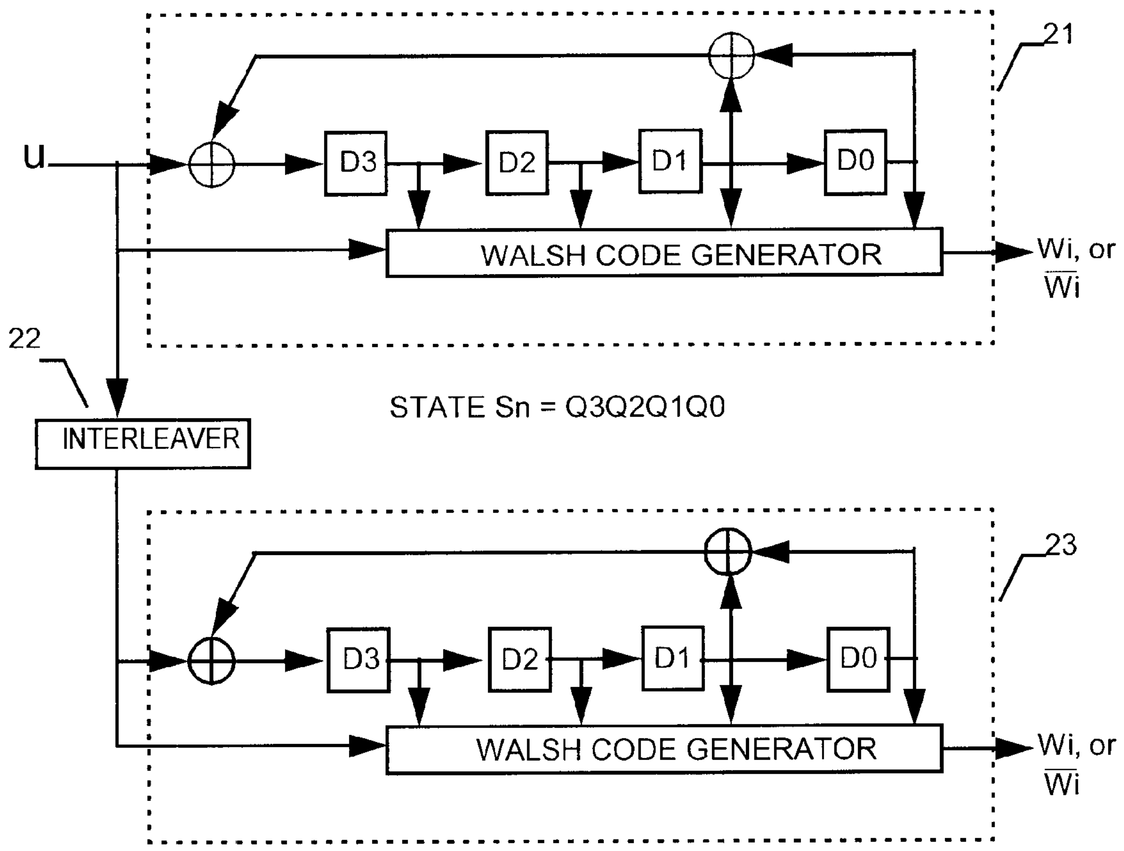
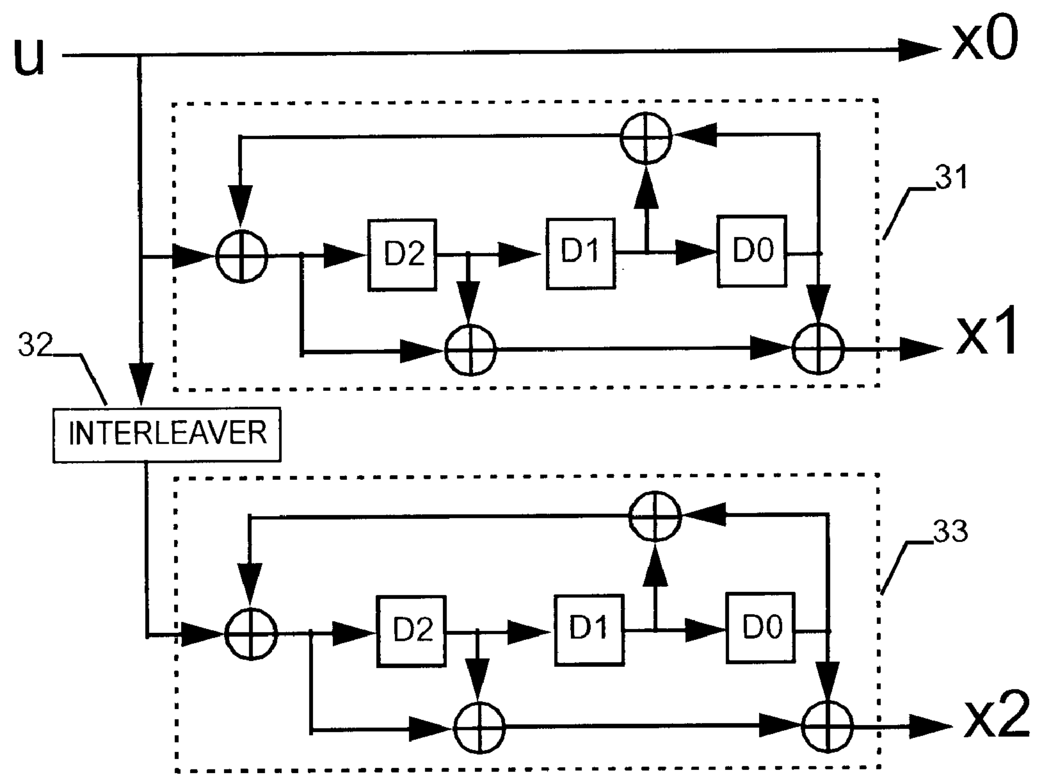


FIGURE 2. A SUPERORTHOGONAL TURBO CODE (SOTC) ENCODER WITH WALSH CODE GENERATOR

(PRIOR ART)



state Sn = Q2Q1Q0

FIGURE 3. THE PARALLEL CONCATENATED CONVOLUTIONAL CODE (PCCC) WITH 8-STATES CONSTITUENT ENCODER, (PRIOR ART)

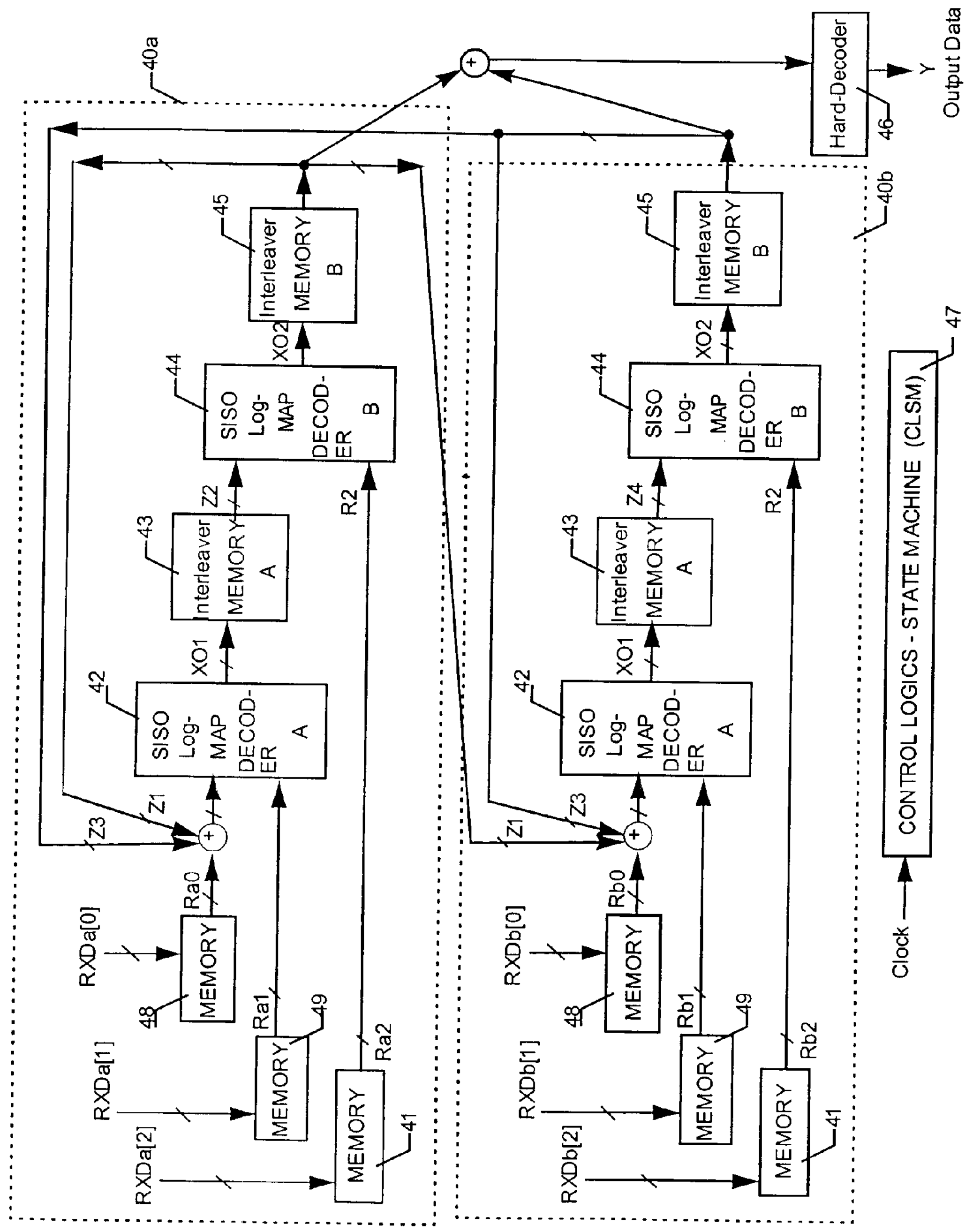


FIGURE 4. TURBO CODES DECODER SYSTEM BLOCK DIAGRAM

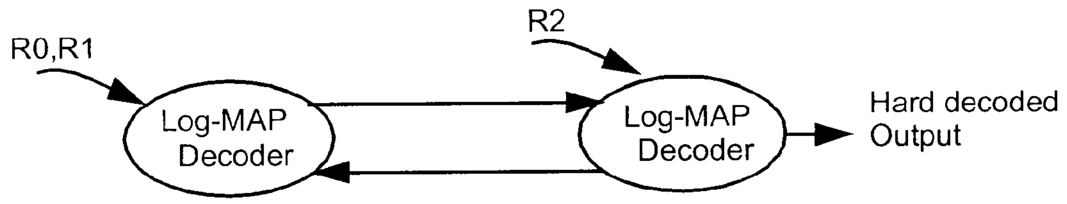


FIGURE 5. TURBO CODES DECODER STATE DIAGRAM

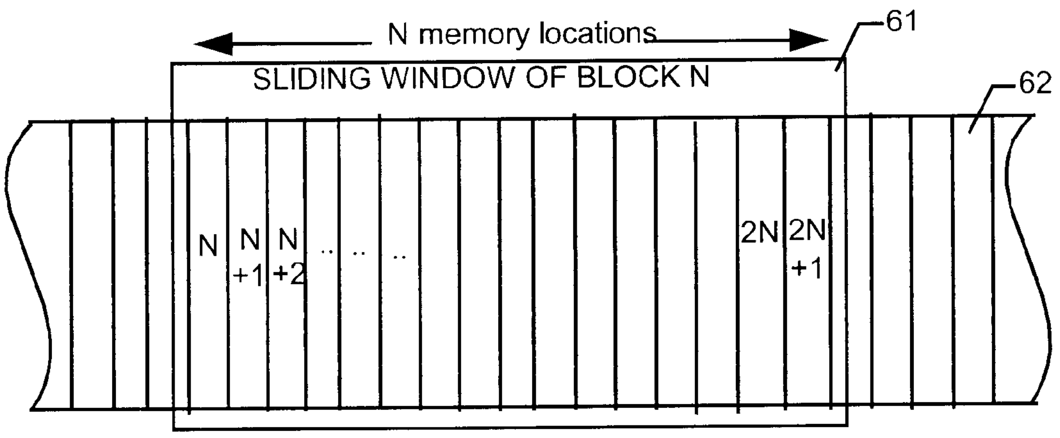


FIGURE 6. BLOCK N SLIDING WINDOW DIAGRAM

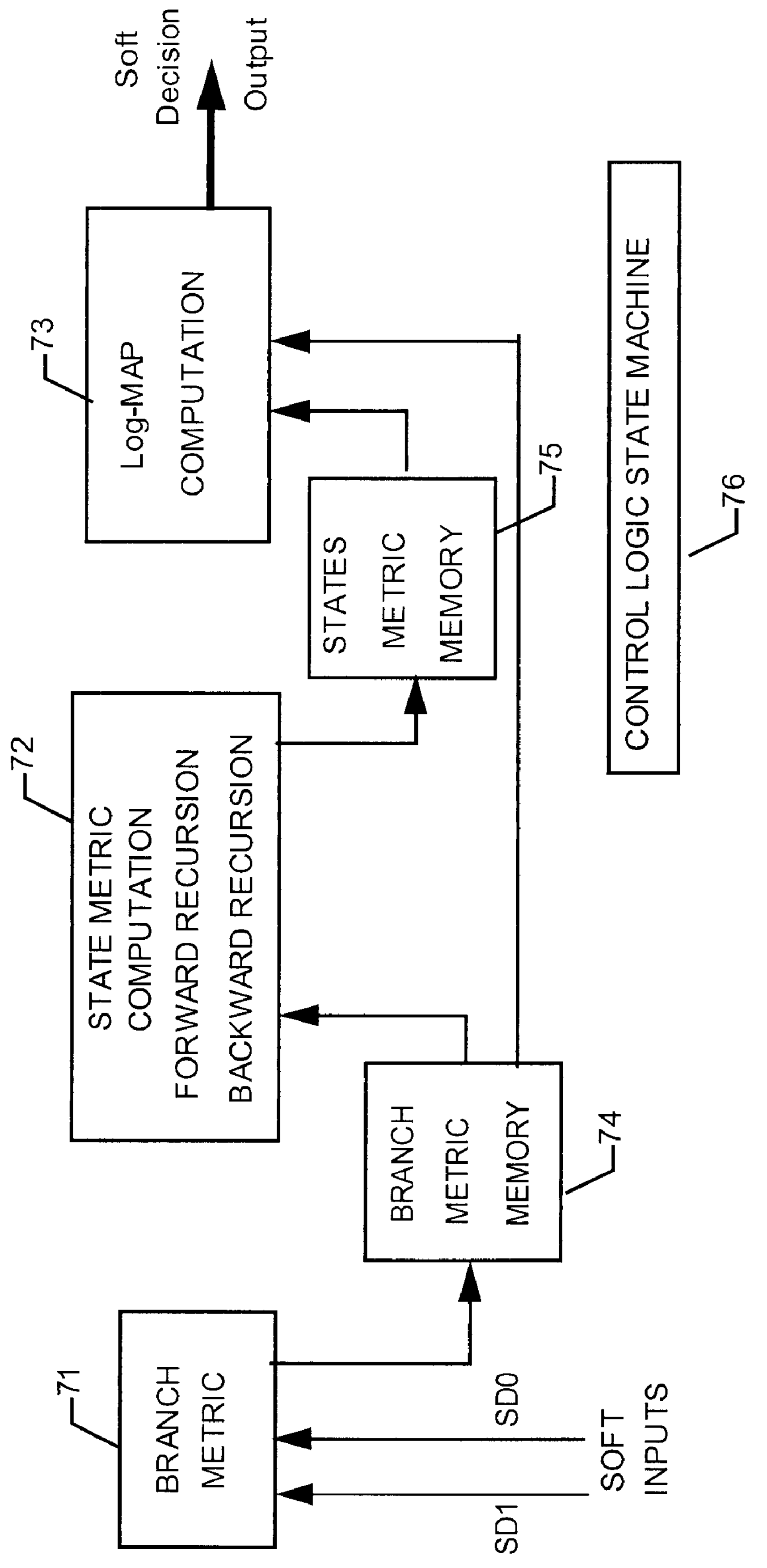


FIGURE 7. THE SISO LOG-MAP DECODER

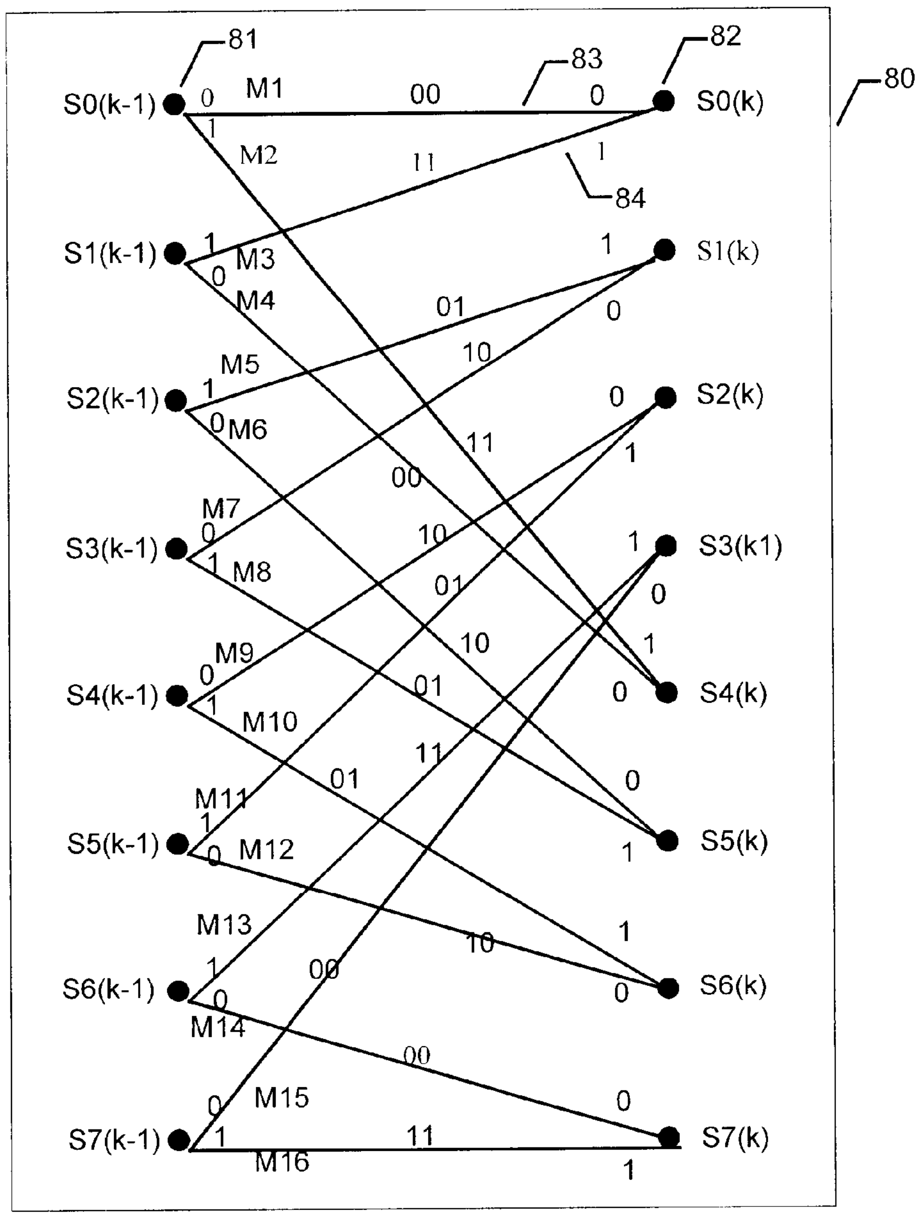


FIGURE 8a. THE 8-STATES TRELLIS DIAGRAM OF A SISO LOG-MAP DECODER

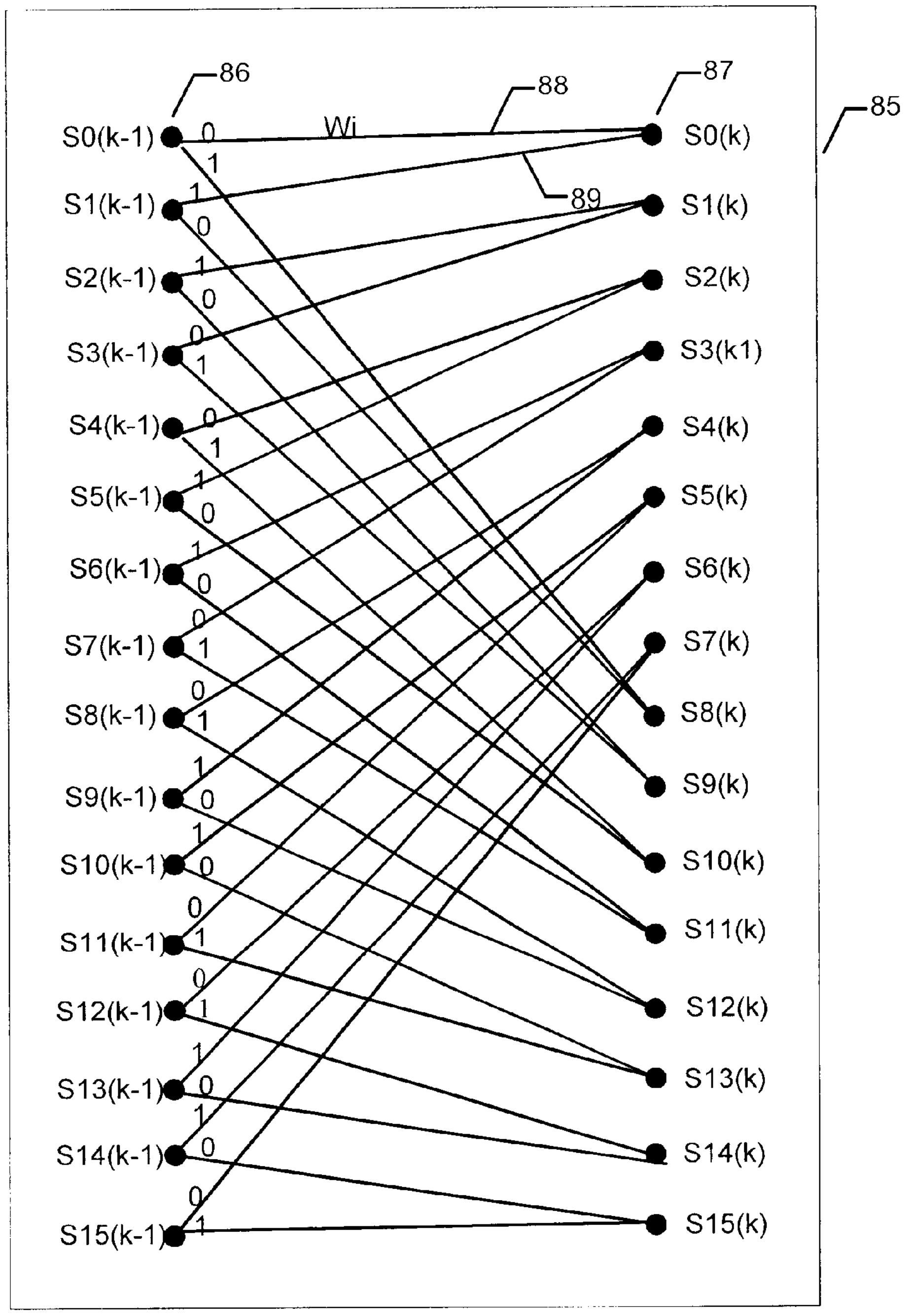


FIGURE 8b. THE 16-STATES TRELLIS DIAGRAM OF A SISO LOG-MAP DECODER

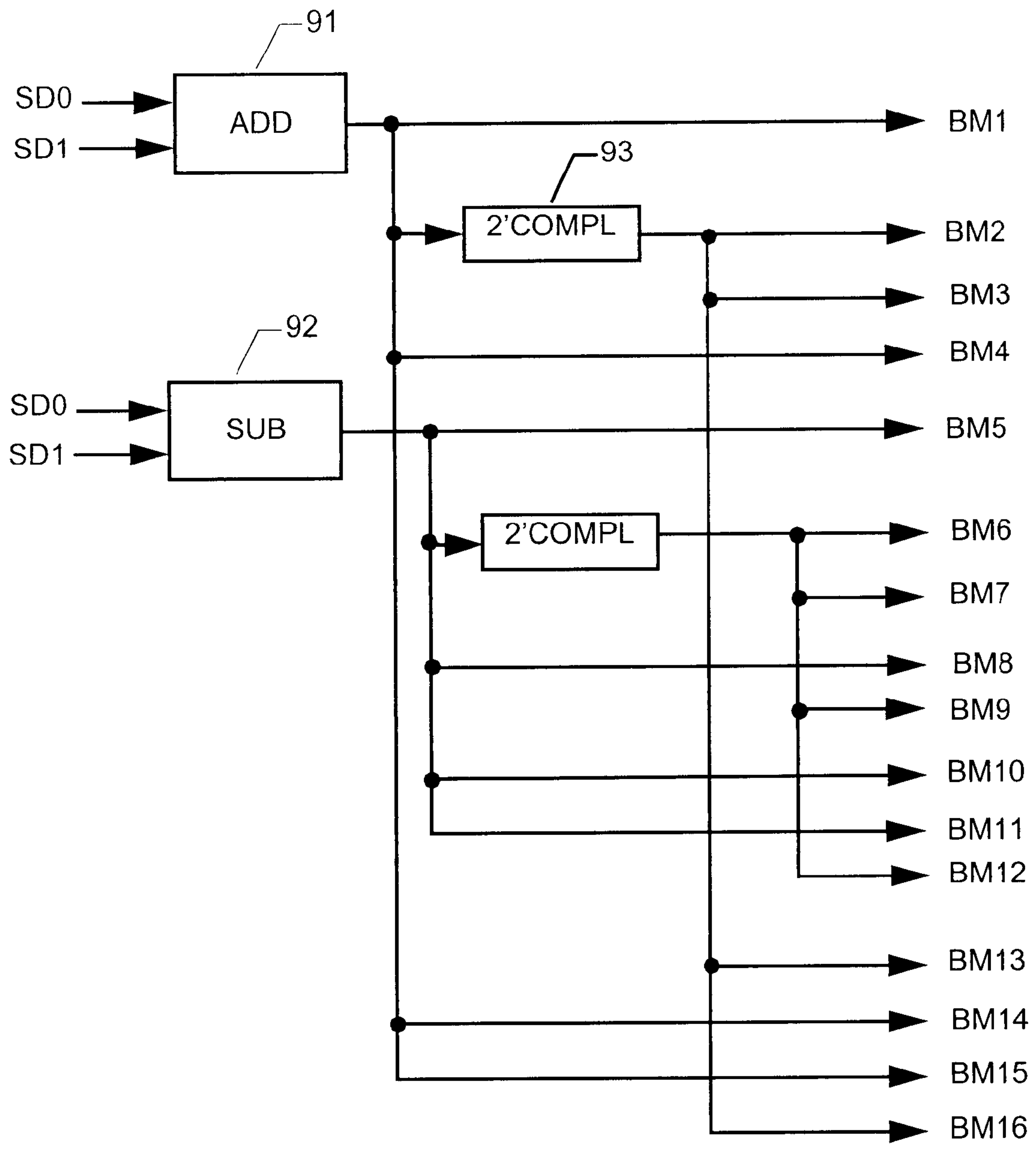


FIGURE 9. BRANCH METRIC COMPUTING MODULE

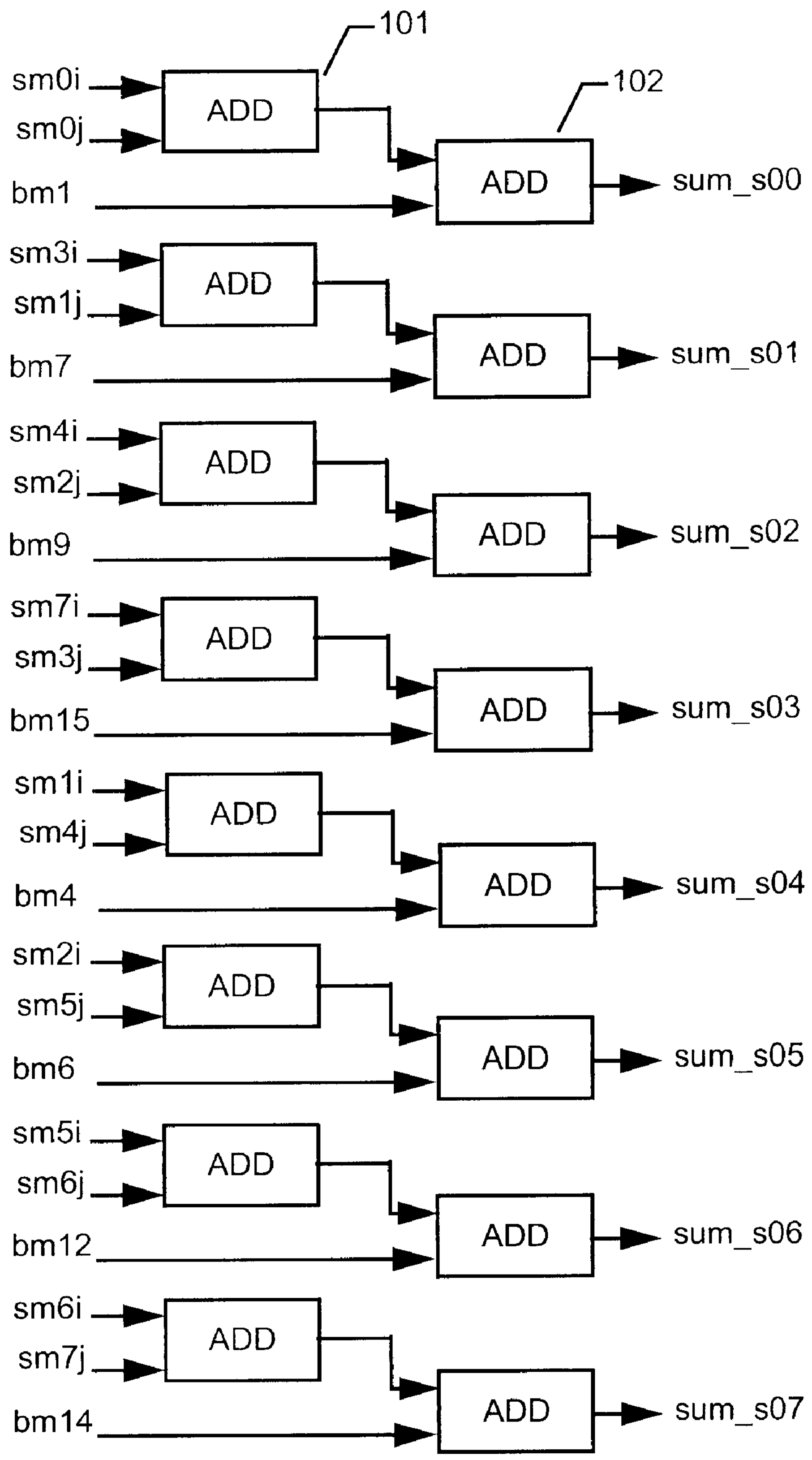


FIGURE 10a. LOG-MAP COMPUTING MODULE

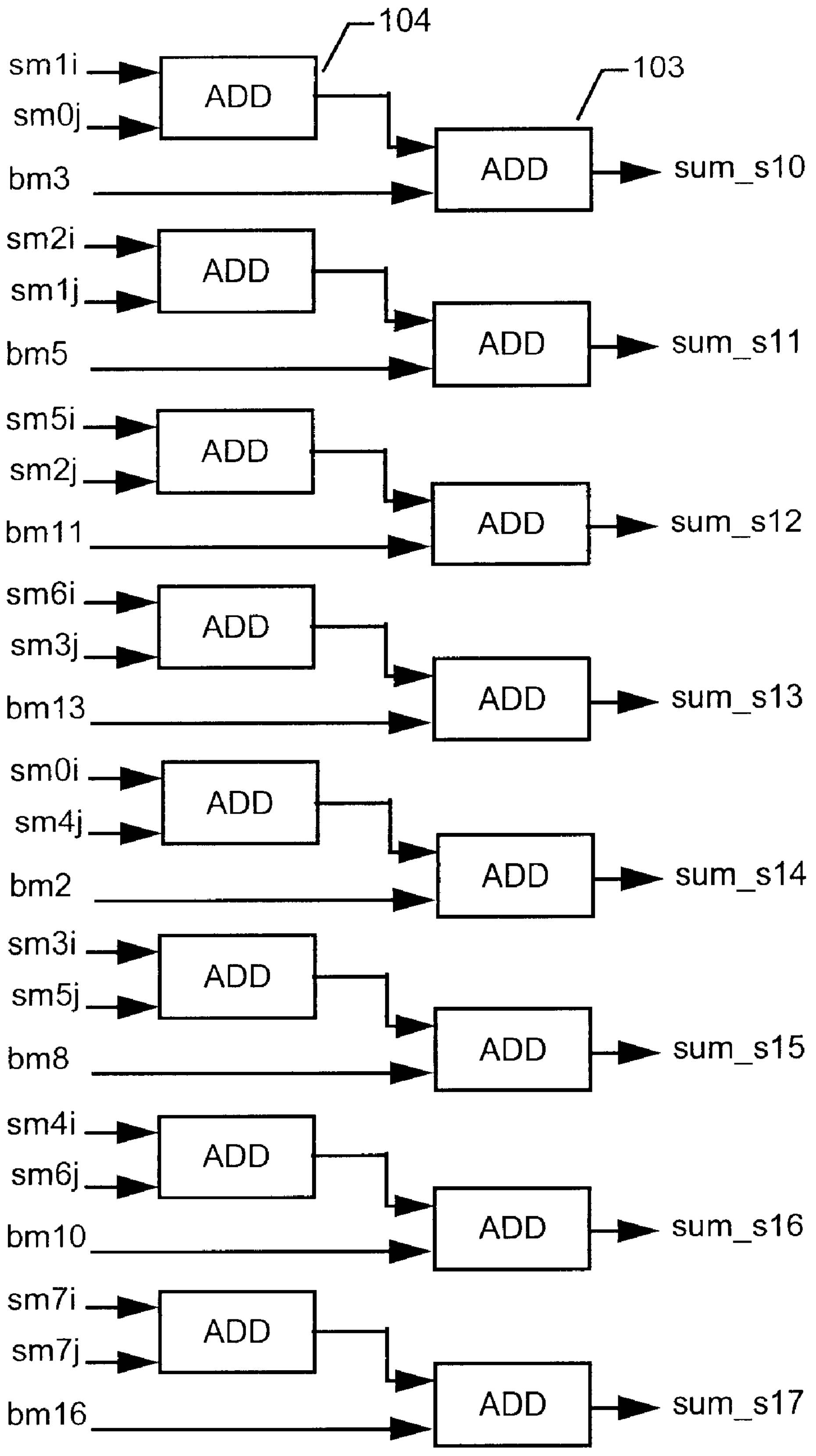


FIGURE 10b. LOG-MAP COMPUTING MODULE

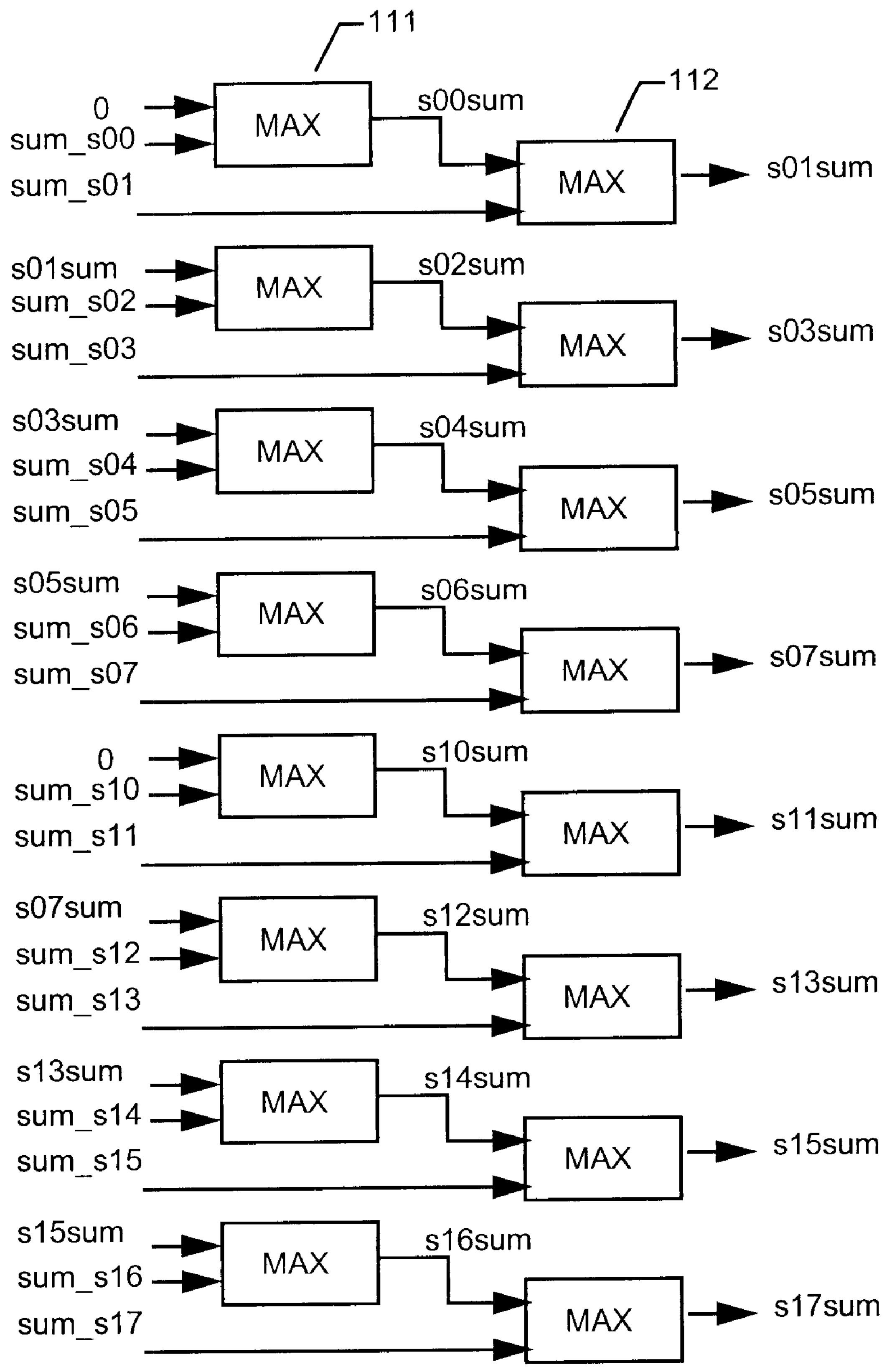


FIGURE 11. COMPUTING LOG-MAP FOR EACH STATE

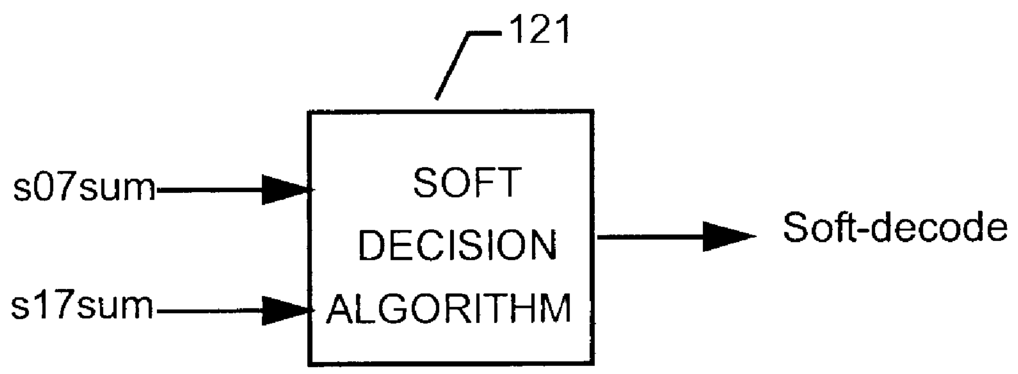


FIGURE 12. SOFT DECODE OUTPUT

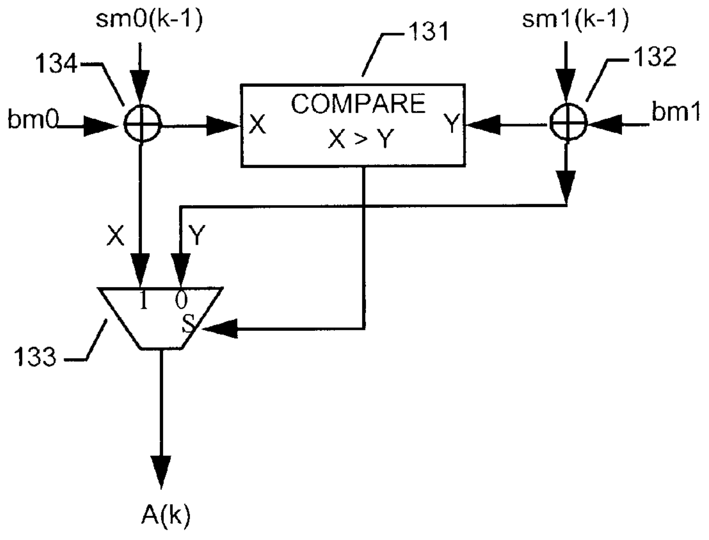


FIGURE 13. COMPUTATION OF FORWARD RECURSION OF STATE-METRIC (ACS)

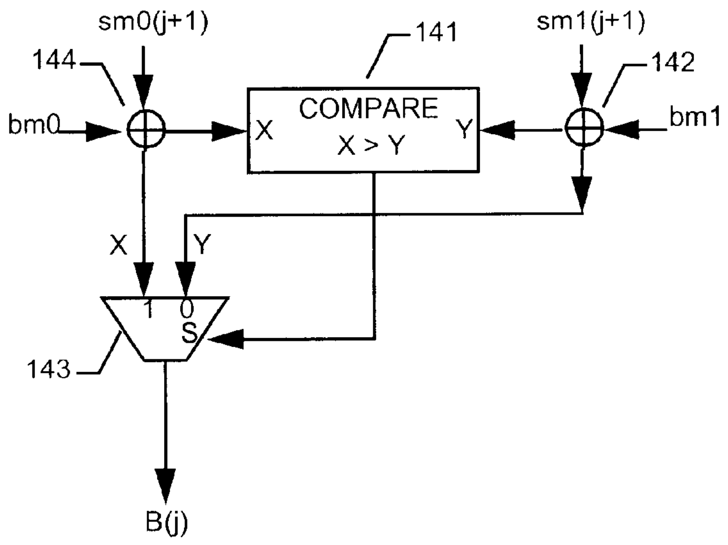


FIGURE 14. COMPUTATION OF BACKWARD RECURSION OF STATE-METRIC (ACS)

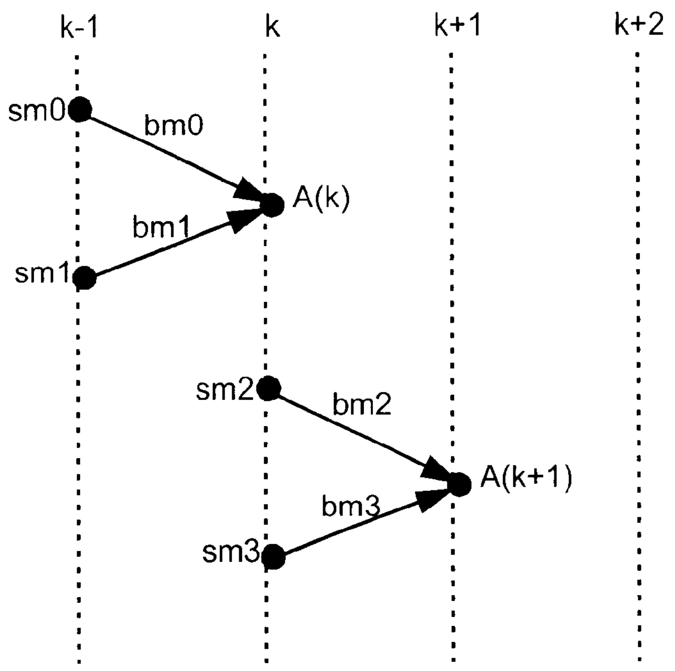


FIGURE 15. FORWARD COMPUTING OF TRELLIS STATE TRANSITIONS

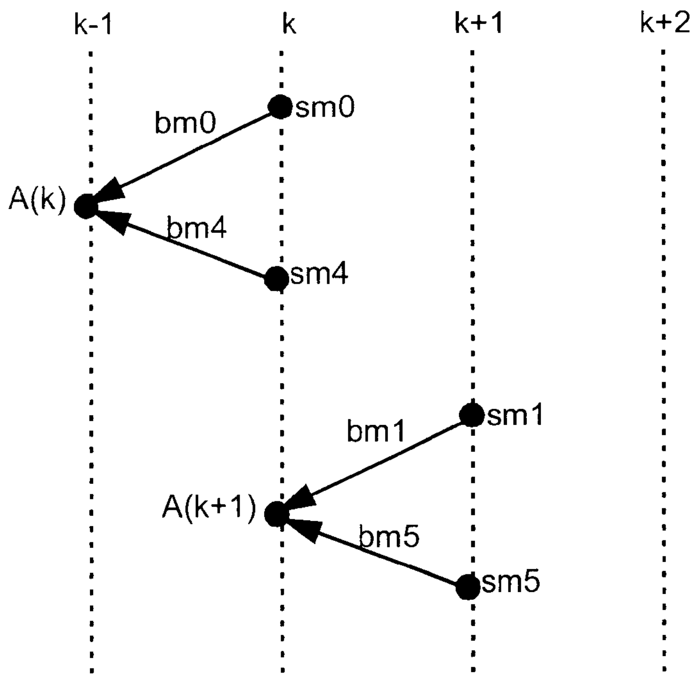


FIGURE 16. BACKWARD COMPUTING OF TRELLIS STATE TRANSITIONS

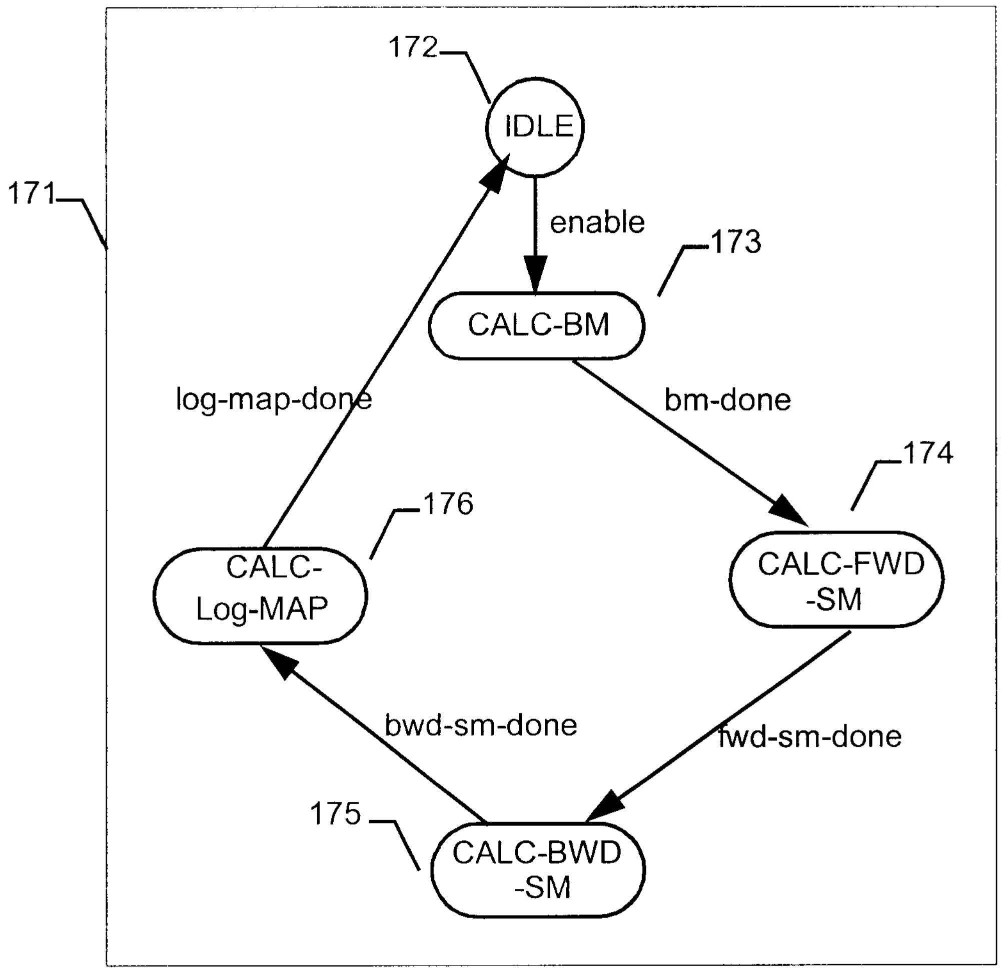


FIGURE 17. STATE MACHINE OPERATIONS OF LOG-MAP DECODER

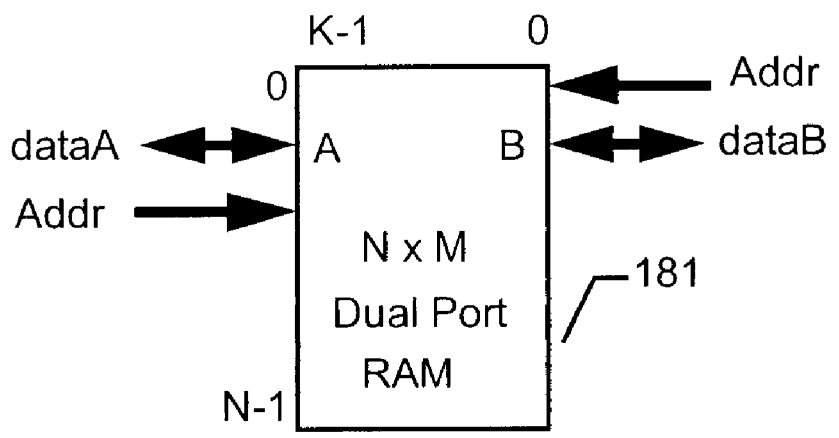


FIGURE 18. BM DUAL-PORT MEMORY MODULE

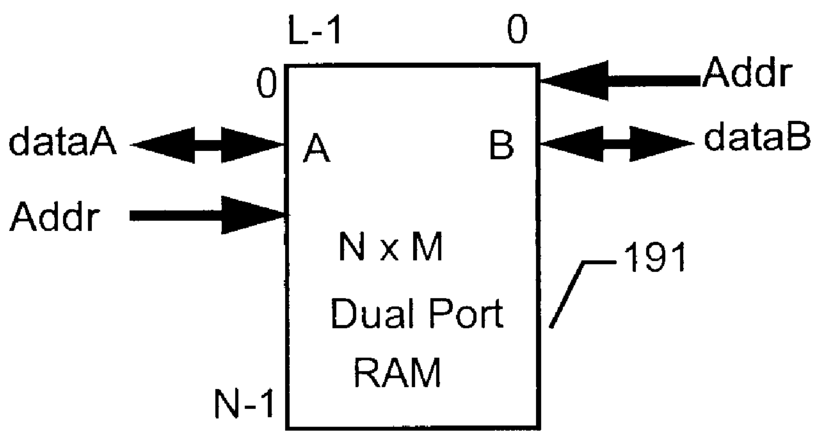


FIGURE 19. SM DUAL-PORT MEMORY MODULE

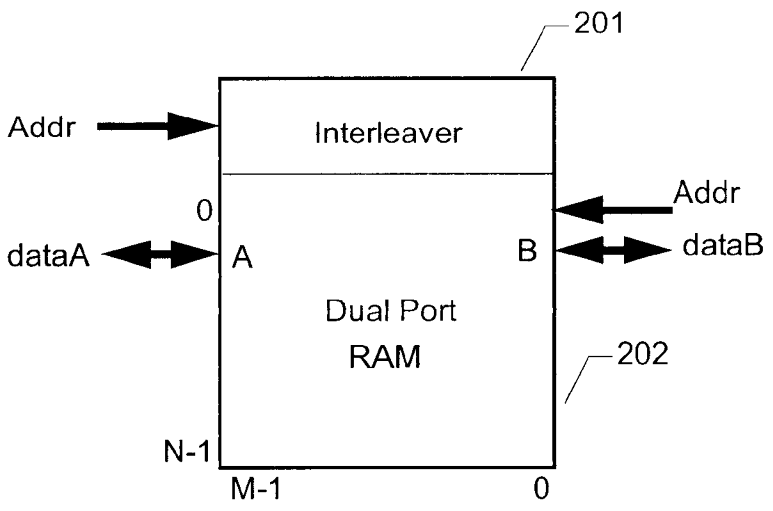


FIGURE 20. INTERLEAVER MEMORY INPUT BUFFER MODULE FOR R2

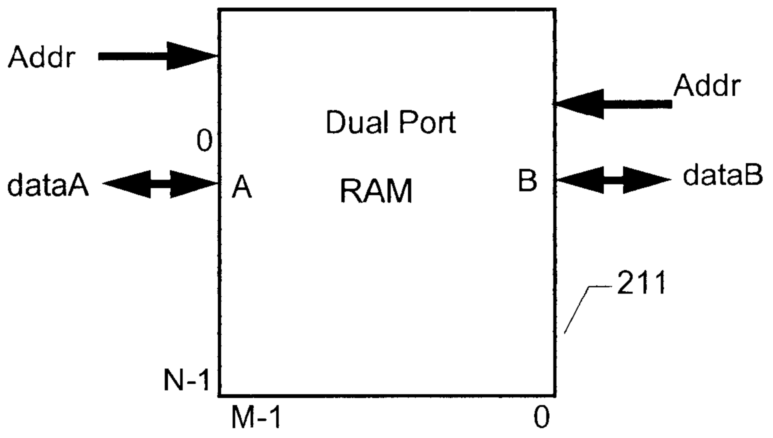


FIGURE 21. MEMORY INPUT BUFFER MODULE FOR R0, R1

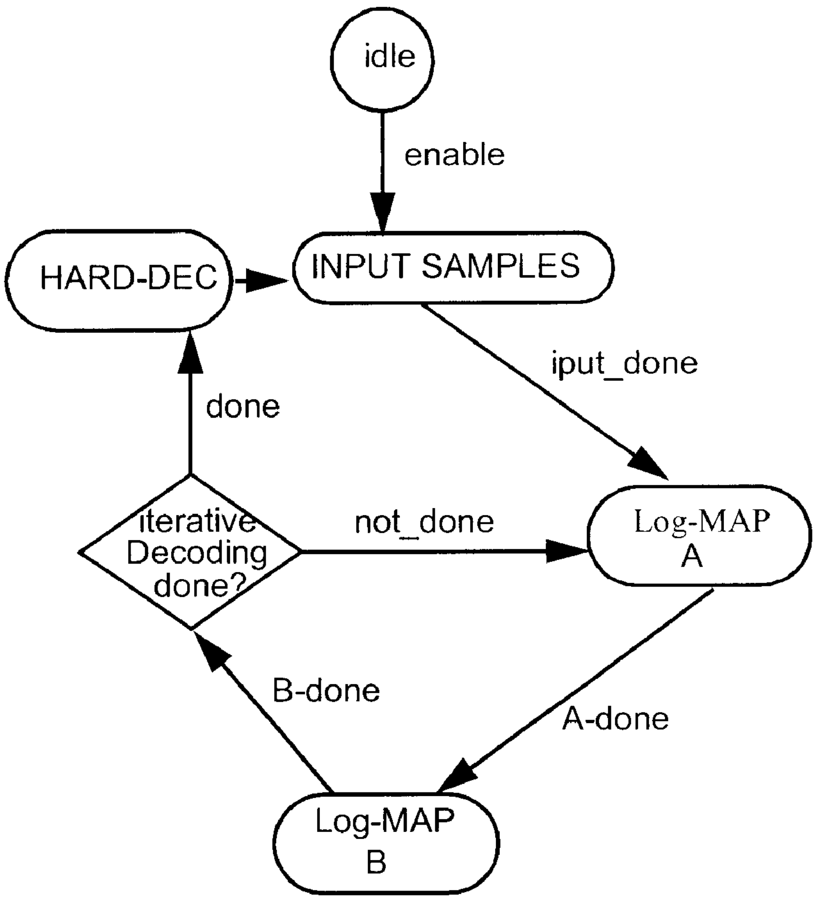


FIGURE 22. STATE MACHINE OPERATIONS OF TURBO DECODER

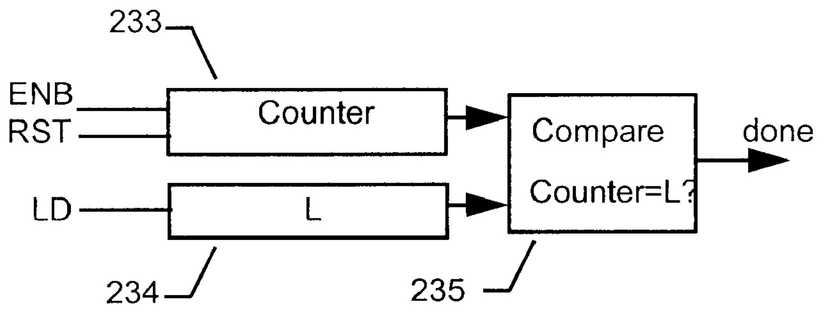


FIGURE 23. ITERATIVE DECODING L TIMES

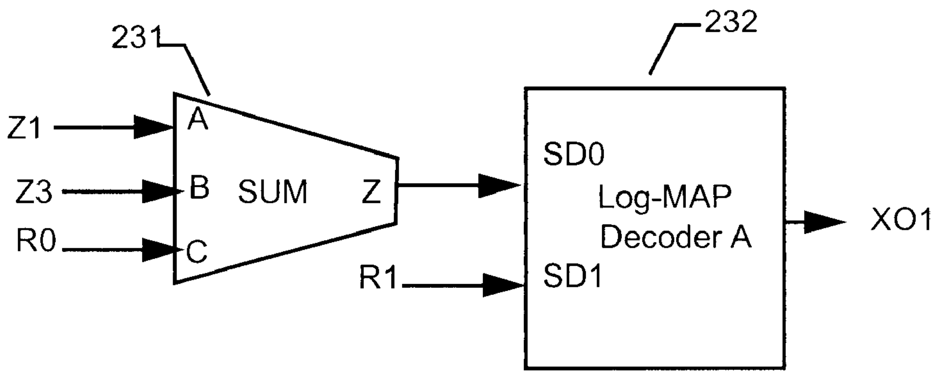


FIGURE 24.
INTRINSIC VALUES FEED-BACK SUM, AND DIVERSITY PROCESSING

HIGH SPEED TURBO CODES DECODER FOR 3G USING PIPELINED SISO LOG-MAP DECODERS ARCHITECTURE

CROSS REFERENCE TO RELATED APPLICATIONS

This application is a continuation-in-part of patent application Ser. No. 10/065408 filed Oct. 15, 2002, which is a continuation-in-part of patent application Ser. No. 09/681093 filed Jan. 2, 2001.

BACKGROUND OF INVENTION

1. Field of the Invention

This invention relates to Wireless Baseband Processors and Forward Error-Correction (FEC) Codes for 3^{rd} Generation (3G) Wireless Mobile Communications More particularly, the invention relates to a very high speed Turbo Codes Decoder using diversity processing and pipelined Max Log-MAP decoders for 3G Code Division Multiple Access (CDMA) 2000 and 3G Wideband Code Division Multiple Access (WCDMA).

2. Description of Prior Art

Diversity processing computes signals from two separate antennas using so-called “multipath” signals that arrive at the terminal via different routes after being reflected from buildings, trees or hills. Diversity processing can increase the signal to noise ratio (SNR) more than 6 dB, which enables 3G systems to deliver data rates up to 2 Mbit/s.

Turbo Codes decoding is based upon the classic forward error correction concepts that include the use of recursive systematic constituent (RSC) Encoders and Interleayers to reduce E_b/N_0 for power-limited wireless applications such as digital 3G Wireless Mobile Communications.

A Turbo Codes Decoder is an important baseband processor of the digital wireless communication Receiver, which was used to reconstruct the corrupted and noisy received data and to improve BER (10^{-6}) throughput. FIG. **1** shows an example of a diversity processing 3G Receiver with a Turbo Codes Decoder **13** which decodes data RXDa and RXDb from Demodulators **11** and Soft Decoders **12**, and sends decoded data to the Media Access Control (MAC) layer **14**. The data from the two received data paths pass through two diversity antennas, two Demodulators **11**, and two Soft Decoders **12** to produce soft decoded data RXDa and RXDb for the Turbo Codes Decoder **13**.

A widely used Forward Error Correction (FEC) scheme is the Viterbi Algorithm Decoder in both wired and wireless applications. A drawback of the Viterbi Algorithm Decoder is that it requires a long wait for decisions until the whole sequence has been received. A delay of six times the memory processing speed of the received data is required for decoding. One of the more effective FEC schemes, with higher complexity, uses a maximum a posteriori (MAP) algorithm to decode received messages. The MAP algorithm is computationally complex, requiring many multiplications and additions per bit to compute the posteriori probability. A major difficulty with the use of the MAP algorithm has been the implementation in semiconductor ASIC devices. The complexity of the multiplications and additions slow down the decoding process and reduce the throughput data rates. Furthermore, even under the best conditions, multiplication operations in the MAP algorithmrequires implementation using large circuits in the ASIC. The result is costly design and low performance in bit rates throughput.

Recently, the 3^{rd} Generation Partnership Project (3GPP) organization introduced a new class of error correction

codes using parallel concatenated codes (PCCC) that include the use of the classic recursive systematic constituent (RSC) Encoders and Interleavers as shown in FIG. **3**. An example of the 3GPP Turbo Codes PCCC with 8-states and rate ⅓ is shown in FIG. **3**. Data enters the two systematic encoders **31 33** separated by an interleaver **32**. An output codeword consists of the source data bit followed by the output bits of the two encoders.

Other prior work relating to error correction codes was performed by Berrou et al., describing parallel concatenated codes which are complex encoding structures that are not suitable for portable wireless device. Another patent U.S. Pat. No. 6,023,783 to Divsalar et al. describes an improved encoding method over Berrou et al., using mathematical concepts of parallel concatenated codes. However, patents by Berrou et al., Divsalar et al., and others only describe the concept of parallel concatenated codes using mathematical equations which are good for research in deep space communications and other government projects, but are not feasible, economical, and suitable for consumer portable wireless devices. In these prior systems, the encoding of data is simple and can be easily implemented with a few xor and flip-flop logic gates. But decoding the Turbo Codes is much more difficult to implement in ASIC or software. The prior art describes briefly the implementation of the Turbo Codes Decoder which are mostly for deep space communications and requires much more hardware, power consumption and costs.

Another prior art example of a 16-state Superorthogonal Turbo Codes (SOTC) is shown in FIG. **2**. It is identical to the previous 3GPP Turbo Codes PCCC except a Walsh Code Generator substitutes for the XOR binary adder. Data enters the two systematic encoders **21**, **23** separated by an interleaver **22**. An output codeword consists of the two Walsh Codes output from the two encoders.

All the prior art Turbo Codes fail to provide simple and suitable methods and architectures for a Turbo Codes Decoder as it is required and desired for 3G cellular phones and 3G personal communication devices, including the features of high speed data throughput, low power consumption, lower costs, limited bandwidth, and limited power transmitter in noisy environments.

SUMMARY OF INVENTION

The present invention is directed to Turbo Code Decoders using diversity processing to implement a more efficient, practical and suitable architecture and method to achieve the requirements for 3G wireless systems, including the features of higher speed data throughput, lower power consumptions, lower costs, and suitable for implementation in ASIC or DSP codes. The present invention encompasses several improved and simplified Turbo Codes Decoder methods and devices to deliver higher speed and lower power consumption, especially for 3G applications. Diversity processing can increase the signal to noise ratio (SNR) more than 6 dB, which enables 3G systems to deliver data rates up to 2 Mbit/s. As shown in FIG. **4**, an exemplary embodiment of the Turbo Codes Decoder utilizes two parallel Turbo Codes Decoders for diversity processing. Each Turbo Codes Decoder has serially concatenated Soft-input Soft-output logarithm maximum a posteriori (SISO Log-MAP) Decoders. The two decoders function in a pipelined scheme with delay latency N. While the first decoder is decoding data stored in the second-decoder-Memory, the second decoder performs decoding for data stored in the first-decoder-Memory, which produces a decoded output every clock cycle. As shown in

FIG. 6, the Turbo Codes Decoder utilizes a Sliding Window of Block N on the input buffer memory to decode data per block N, which improves processing efficiency. Accordingly, several objects and advantages of the Turbo Codes Decoder are:

To implement diversity processing to increase the signal to noise ratio (SNR).

To deliver higher speed throughput and be suitable for implementation in application specific integrated circuit (ASIC) designs or digital signal processor (DSP) codes.

To utilize SISO Log-MAP decoders for faster decoding and simplified implementation in ASIC circuits and DSP codes with the use of binary adders for computation.

To perform re-iterative decoding of data back-and-forth between the two Log-MAP decoders in a pipelined scheme until a decision is made. In such pipelined scheme, decoded output data is produced each clock cycle.

To utilize a Sliding Window of Block N on the input buffer memory to decode data per block N for improved pipeline processing efficiency

To provide higher performance in term of symbol error probability and low BER (10^{-6}) for 3G applications such as 3G W-CDMA, and 3G CDMA2000 operating at very high bit-rate up to 100 Mbps, in a low power, noisy environment.

To utilize a simplified and improved SISO Log-MAP decoder architecture, including a branch-metric (BM) calculations module, a recursive state-metric (SM) forward/backward calculations module, an Add-Compare-Select (ACS) circuit, a Log-MAP posteriori probability calculations module, and an output decision module.

To reduce complexity of multiplier circuits in MAP algorithm by performing the entire MAP algorithm in Log Max approximation using binary adder circuits, which are more suitable for ASIC and DSP codes implementation, while still maintaining a high level of performance output.

To design an improve Log-MAP Decoder using high level design language (HDL) such as Verilog, system-C and VHDL, which can be synthesized into custom ASIC and Field Programmable Gate Array (FPGA) devices.

To implement an improve Log-MAP Decoder in DSP (digital signal processor) using optimized high level language C, C++, or assembly language.

Still further objects and advantages will become apparent to one skill in the art from a consideration of the ensuing descriptions and accompanying drawings.

BRIEF DESCRIPTION OF DRAWINGS

FIG. **1** illustrates a conventional 3G Receiver Functional Block Diagram which uses Turbo Codes Decoder for error-correction.

FIG. **2** illustrates an example of a conventional 16-states Superorthogonal Turbo Code (SOTC) Encoder with Walsh code generator.

FIG. **3** illustrates a block diagram of a conventional 8-states 3GPP Parallel Concatenated Convolutional Codes.

FIG. **4** illustrates the Turbo Codes Decoder System Block Diagram showing Log-MAP Decoders, Interleavers, Memory Buffers, and control logics.

FIG. **5** illustrates a Turbo Codes Decoder State Diagram.

FIG. **6** illustrates the Block N Sliding Window Diagram.

FIG. **7** illustrates a block diagram of the SISO Log-MAP Decoder showing Branch Metric module, State Metric module, Log-MAP module, and State and Branch Memory modules.

FIG. **8***a* illustrates the 8-States Trellis Diagram of a SISO Log-MAP Decoder using the 3GPP 8-state PCCC Turbo codes.

FIG. **8***b* illustrates the 16-States Trellis Diagram of a SISO Log-MAP Decoder using the superorthogonal Turbo codes (SOTC).

FIG. **9** illustrates a block diagram of the BRANCH METRIC COMPUTING module.

FIG. **10***a* illustrates a block diagram of the Log-MAP computing for u=0.

FIG. **10***b* illustrates a block diagram of the Log-MAP computing for u=1.

FIG. **11** illustrates a block diagram of the Log-MAP Compare & Select 1 maximum logic for each state.

FIG. **12** illustrates a block diagram of the Soft Decode module.

FIG. **13** illustrates a block diagram of the Computation of Forward Recursion of State Metric module (FACS).

FIG. **14** illustrates a block diagram of the Computation of Backward Recursion of State Metric module (BACS).

FIG. **15** illustrates State Metric Forward computing of Trellis state transitions.

FIG. **16** illustrates State Metric Backward computing of Trellis state transitions.

FIG. **17** illustrates a block diagram of the State Machine operations of Log-MAP Decoder.

FIG. **18** illustrates a block diagram of the BM dual-port Memory Module.

FIG. **19** illustrates a block diagram of the SM dual-port Memory Module.

FIG. **20** illustrates a block diagram of the De-Interleaver dual-port RAM Memory Memory Module for interleaved input R**2**.

FIG. **21** illustrates a block diagram of the dual RAM Memory Module for input R**0**,R**1**.

FIG. **22** illustrates a flow chart of an exemplary state machine operation.

FIG. **23** illustrates a block diagram of the Iterative decoding feedback control.

FIG. **24** illustrates a block diagram of the intrinsic feedback Adder of the Turbo Codes Decoder.

DETAILED DESCRIPTION

Turbo Codes Decoder

An illustration of a 3GPP 8-state Parallel Concatenated Convolutional Code (PCCC), with coding rate ⅓, constraint length K=4 is illustrated in FIG. **3**. An implementation using SISO Log-MAP Decoders is illustrated in FIG. **4**.

In accordance with an exemplary embodiment, a diversity processing Turbo Codes Decoder includes two parallel blocks **40***a*, **40***b* of Turbo Codes Decoders for each path of received data RXDa and RXDb. Each identical Turbo Codes Decoder block **40***a*, **40***b* has concatenated max Log-MAP SISO Decoders A **42** and B **44** connected in a feedback loop with Interleaver Memory **43** and Interleaver Memory **45**. The Soft output of Turbo Codes Decoder block **40***a* is fed-back into the input of Turbo Codes Decoder block **40***b*. Conversely, the Soft output of Turbo Codes Decoder block **40***b* is fed-back into the input of Turbo Codes Decoder block **40***a*. The sum of the two outputs Z**1**, Z**3** of the Turbo Codes Decoder block **40***a*, **40***b* is fed into the Hard-Decoder to generate output Y data.

Signals Ra**2**, Ra**1**, Ra**0** are received soft decision signals of data path A from the system receiver. Signals XO1 and XO2 are output soft decision signals of the Log-MAP Decoders A **42** and B **44**, respectively, which are stored in the Interleaver Memory **43** and Memory **45** module. Signals Z2 and Z1 are the output of the Interleaver Memory **43** and Interleaver Memory **45**. Z2 is fed into Log-MAP decoder B **44** and Z1 is looped back into Log-MAP decoder A **42** through Adder **231**.

Signals Rb2, Rb1, Rb**0** are received soft decision signals of data path B from the system receiver. Signals XO1 and XO2 are output soft decision of the Log-MAP Decoders A **42** and B **44**, respectively, which are stored in the Interleaver Memory **43** and Memory **45** module. Signals Z4 and Z3 are the output of the Interleaver Memory **43** and Interleaver Memory **45**. Z4 is fed into Log-MAP decoder B **44** and Z3 is looped back into Log-MAP decoder A **42** through Adder **231**.

In accordance with the invention, signal Z3 is fed back into Log-MAP decoder A **42** of block **40***a* through Adder **231**, and Signal Z1 is fed back into Log-MAP decoder A **42** of block **40***b* through Adder **231** for diversity processing.

Each Interleaver Memory **43**, **45**, shown in FIG. **20**, includes one interleaver **201** and a dual-port RAM memory **202**. Input Memory blocks **41**, **48**, **49**, shown in FIG. **21**, include dual-port RAM memory **211**. Control logic module (CLSM) **47** consists of various state-machines, which control all the operations of the Turbo Codes Decoder. The hard-decoder module **46** outputs the final decoded data.

More particularly, as illustrated in FIG. **3**, Ra**0**, Rb**0** are data bits corresponding to the transmit data bit u, Ra**1**, Rb**1** are the first parity bits corresponding to the output bit of the first RSC encoder, and Ra**2**, Rb**2** are interleaved second parity bits corresponding to the output bit of the second RSC encoder.

In accordance with the invention, corresponding ones of data bits Ra**0**, Rb**0** are added to the feedback signals Z1 and Z3, then fed into the decoder A. Corresponding ones of data bits Ra**1**, Rb**1** are also fed into decoder A for decoding the first stage of decoding output X**01**. Z2 and corresponding ones of Ra**2**, Rb**2** are fed into decoder B for decoding the second stage of decoding output X**02**.

In accordance with the invention, as shown in FIG. **6**, the Turbo Codes Decoder utilizes a Sliding Window of Block N **61** on the input buffers **62** to decode one block N data at a time, the next block N of data is decoded after the previous block N is done in a circular wrap-around scheme for pipeline operations. In another embodiment, the Sliding Window of Block N is used on the input buffer Memory so that each block N data is decoded at a time one block after another in a pipeline scheme.

In accordance with the invention, the Turbo Codes Decoder decodes an 8-state Parallel Concatenated Convolutional Code (PCCC), and also decodes a 16-states Super-orthogonal Turbo Codes SOTC with different code rates. The Turbo Codes Decoder also decodes a higher n-state Parallel Concatenated Convolutional Code (PCCC)

As illustrated in FIG. **4**, the Turbo Codes Decoder functions effectively as follows:

Received soft decision data (RXDa[2:0]) is stored in three input buffers Memorys **48**, **49**, **41** to produce data bits Ra**0**, Ra**1**, and Ra**2** that correspond to data words. Each output data word Ra**0**, Ra**1**, Ra**2** contains a number of binary bits.

Received soft decision data (RXDb[2:0]) is stored in three input buffers Memorys **48**, **49**, **41** to produce Rb**0**, Rb**1**, and Rb**2** that correspond to data words. Each output data word Rb**0**, Rb**1**, Rb**2** contains a number of binary bits.

A Sliding Window of Block N is imposed onto each input memory to produce corresponding ones of Ra**0**, Rb**0**, Ra**1**, Rb**1**, Ra**2**, and Rb**2** output data words.

In accordance with the method of the invention, when an inpuot data block of size N is ready, the Turbo Decoder starts the Log-MAP Decoder A, in block **40***a*, to decode the N input data based on the soft-values of Ra**0**, Z1, Z3 and Ra**1**, then stores the outputs in the Interleaver Memory A.

The Turbo Decoder also starts the Log-MAP Decoder B, in block **40***a*, to decode the N input data based on the soft-values of Ra**2** and Z2, in pipelined mode with a delay latency of N, then stores the output in the Interleaver Memory.

When an input data block of size N is ready, the Turbo Decoder starts the Log-MAP Decoder A, in block **40***b*, to decode the N input data based on the soft-values of Rb**0**, Z1, Z3 and Rb**1**, then stores the outputs in the Interleaver Memory A.

The Turbo Decoder also starts the Log-MAP Decoder B, in block **40***b*, to decode the N input data based on the soft-values of Rb**2** and Z4, in pipelined mode with a delay latency of N, then store the outputs in the Interleaver Memory.

The Turbo Decoder performs iterative decoding for L number of times (L=1,2, . . . , M). The Log-MAP Decoder A receives the sum of (Z1 and Z3 and corresponding ones of Ra**0**, Rb**0** as inputs. The Log-MAP Decoder A also receives corresponding ones of Ra**1**, Rb**1** as inputs. The Log-MAP Decoder B receives the data Z2 and R**2** as inputs.

When the iterative decoding sequences is complete, the Turbo Decoder starts the hard-decision operations to compute and produce soft-decision outputs.

Sisco Log-map Decoder

As shown in FIG. **7**, SISO Log-MAP Decoders **42**, **44** include a Branch Metric (BM) computation module **71**, a State Metric (SM) computation module **72**, a Log-MAP computation module **73**, a BM Memory module **74**, a SM Memory module **75**, and a Control Logic State Machine module **76**. Soft-value inputs enter the Branch Metric (BM) computation module **71**, where Euclidean distance is calculated for each branch, the output branch metrics are stored in the BM Memory module **74**. The State Metric (SM) computation module **72** reads branch metrics from the BM Memory **74** and computes the state metric for each state, the output state-metrics are stored in the SM Memory module **75**. The Log-MAP computation module **73** reads both branch-metrics and state-metrics from BM memory **74** and SM memory **75** modules to compute the Log Maximum a Posteriori probability and produce soft-decision output. The Control Logic State-machine module **76** provides the overall operations of the decoding process.

As shown in FIG. **7** which is one example of 3GPP Turbo Codes Decoder, the Log-MAP Decoder **42 44** functions effectively as follows:

The Log-MAP Decoder **42**, **44** reads each soft-values (SD) data pair input, then computes branch-metric (BM) values for all paths in the Turbo Codes Trellis **80** as shown in FIG. **8***a* (and Trellis **85** in FIG. **8***b*). The computed BM data is stored into BM Memory **74**. The process of computing BM values is repeated for each input data until all N samples are calculated and stored in BM Memory **74**.

The Log-MAP Decoder **42 44** reads BM values from BM Memory **74** and SM values from SM Memory **75**, and computes the forward state-metric (SM) for all states in the Trellis **80** as shown in FIG. **8***a* (and Trellis **85** in FIG. **8***b*). The computed forward SM data is stored into SM Memory **75**. The process of computing forward SM values is repeated for each input data until all N samples are calculated and stored in SM Memory **75**.

The Log-MAP Decoder **42 44** reads BM values from BM Memory **74** and SM values from SM Memory **75**, and computes the backward state-metric (SM) for all states in the Trellis **80** as shown in FIG. **8***a* (and Trellis **85** in FIG. **8***b*). The computed backward SM data is stored into the SM Memory **75**. The process of computing backward SM values is repeated for each input data until all N samples are calculated and stored in SM Memory **75**.

The Log-MAP Decoder **42 44** then computes Log-MAP posteriori probability for u=0 and u=1 using the BM values and SM values from BM Memory **74** and SM Memory **75**. The process of computing Log-MAP posteriori probability is repeated for each input data until all N samples are calculated. The Log-MAP Decoder then decodes data by making soft decision based on the posteriori probability for each stage and produces soft-decision output, until all N inputs are decoded.

Branch Metric Computation Module

The Branch Metric (BM) computation module **71** computes the Euclidean distance for each branch in the 8-states Trellis **80** as shown in the FIG. **8***a* based on the following equations:

Local Euclidean distances values=SD**0***G**0**+SD**1***G**1**

where SD**0** and SD**1** are soft-value input data and G**0** and G**1** are the expected input for each path in the Trellis **80**. G**0** and G**1** are coded as signed antipodal values, meaning that 0 corresponds to +1 and 1 corresponds to −1. Therefore, the local Euclidean distances for each path in the Trellis **80** are computed by the following equations:

M**1**=SD**0**+SD**1**
M**2**=−M**1**
M**3**=M**2**
M**4**=M**1**
M**5**=−SD**0**+SD**1**
M**6**=−M**5**
M**7**=M**6**
M**8**=M**5**
M**9**=M**6**
M**10**=M**5**
M**11**=M**5**
M**12**=M**6**
M**13**=M**2**
M**14**=M**1**
M**15**=M**1**
M**16**=M**2**

As shown in the exemplary embodiment of FIG. **9**, the Branch Metric Computing module includes one L-bit Adder **91**, one L-bit Subtracter **92**, and a 2'complemeter **93**. The Euclidean distances is computed for path M**1** and M**5**. Path M**2** is 2'complement of path M**1**. Path M**6** is 2'complement of M**5**. Path M**3** is the same path M**2**, path M**4** is the same as path M**1**, path M**7** is the same as path M**6**, path M**8** is the same as path M**5**, path M**9** is the same as path M**6**, path M**10** is the same as path M**5**, path M**11** is the same as path M**5**, path M**12** is the same as path M**6**, path M**13** is the same as path M**2**, path M**14** is the same as path M**1**, path M**15** is the same as path M**1**, and path M**16** is the same as path M**2**.

State Metric Computing Module

The State Metric Computing module **72** calculates the probability A(k) of each state transition in forward recursion and the probability B(k) in backward recursion. FIG. **13** shows the implementation of state-metric in forward recursion with Add-Compare-Select (ACS) logic. FIG. **14** shows the implementation of state-metric in backward recursion with Add-Compare-Select (ACS) logic. The calculations are performed at each node in the Turbo Codes Trellis **80** (FIG. **8***a*) in both forward and backward recursion. FIG. **15** shows the forward state transitions in the Turbo Codes Trellis **80** (FIG. **8***a*). FIG. **16** shows the backward state transitions in the Turbo Codes Trellis **80** (FIG. **8***a*). Each node in the Trellis **80** as shown in FIG. **8***a* has two entering paths: one-path **84** and zero-path **83**, from the two nodes in the previous stage.

In an exemplary embodiment, the ACS logic includes an Adder **132**, an Adder **134**, a Comparator **131**, and a Multiplexer **133**. In the forward recursion, the Adder **132** computes the sum of the branch metric and state metric in the one-path **84** from the state s(k−1) of previous stage (k−1). The Adder **134** computes the sum of the branch metric and state metric in the zero-path **83** from the state (k−1) of previous stage (k−1). The Comparator **131** compares the two sums and the Multiplexer **133** selects the larger sum for the state s(k) of current stage (k). In the backward recursion, the Adder **142** computes the sum of the branch metric and state metric in the one-path **84** from the state s(j+1) of previous stage (J+1). The Adder **144** computes the sum of the branch metric and state metric in the zero-path **83** from the state s(j+1) of previous stage (J+1). The Comparator **141** compares the two sums and the Multiplexer **143** selects the larger sum for the state s(j) of current stage (j).

The Equations for the ACS are shown below:

A(k)=MAX [(bm**0**+sm**0**(k−1)), (bm**1**+sm**1**(k−1)]
B(j)=MAX [(bm**0**+sm**0**(j+1)), (bm**1**+sm**1**(j+1)]

Time (k−1) is the previous stage of (k) in forward recursion as shown in FIG. **15**, and time (j+1) is the previous stage of (j) in backward recursion as shown in FIG. **16**.

Log-MAP Computing Module

The Log-MAP computing module calculates the posteriori probability for u=0 and u=1, for each path entering each state in the Turbo Codes Trellis **80** corresponding to u=0 and u=1 or referred as zero-path **83** and one-path **84**. The accumulated probabilities are compared and the u with larger probability is selected. The soft-decisions are made based on the final probability selected for each bit. FIG. **10***a* shows the implementation for calculating the posteriori probability for u=0. FIG. **10***b* shows the implementation for calculating the posteriori probability for u=1. FIG. **11** shows the implementation of compare-and-select for the u with larger probability. FIG. **12** shows the implementation of the soft-decode compare logic to produce output bits based on the posteriori probability of u=0 and u=1. The equations for calculating the accumulated probabilities for each state and compare-and-select are shown below:

sum_s**00**=sm**0***i*+bm**1**+sm**0***j*
sum_s**01**=sm**3***i*+bm**7**+sm**1***j*
sum_s**02**=sm**4***i*+bm**9**+sm**2***j*
sum_s**03**=sm**7***i*+bm**15**+sm**3***j*
sum_s**04**=sm**1***i*+bm**4**+sm**4***j*
sum_s**05**=sm**2***i*+bm**6**+sm**5***j*
sum_s**06**=sm**5***i*+bm**12**+sm**6***j*
sum_s**07**=sm**6***i*+bm**14**+sm**7***j*
sum_s**10**=sm**1***i*+bm**3**+sm**0***j*
sum_s**11**=sm**2***i*+bm**5**+sm**1***j*
sum_s**12**=sm**5***i*+bm**11**+sm**2***j*
sum_s**13**=sm**6***i*+bm**13**+sm**3***j*
sum_s**14**=sm**0***i*+bm**2**+sm**4***j*

sum__s**15**=sm**3***i*+bm**8**+sm**5***j*
sum__s**16**=sm**4***i*+bm**10**+sm**6***j*
sum__s**17**=sm**7***i*+bm**16**+sm**7***j*
s**00**sum=MAX[sum__s**00**, **0**]
s**01**sum=MAX[sum__s**01**, s**00**sum]
s**02**sum=MAX[sum__s**02**, s**01** sum]
s**03**sum=MAX[sum__s**03**, s**02**sum]
s**04**sum=MAX[sum__s**04**, s**03**sum]
s**05**sum=MAX[sum__s**05**, s**04**sum]
s**06**sum=MAX[sum__s**06**, s**05**sum]
s**07**sum=MAX[sum__s**07**, s**06**sum]
s**10**sum=MAX[sum__s**10**, **0**]
s**11**sum=MAX[sum__s**11**, s**10**sum]
s**12**sum=MAX[sum__s**12**, s**11**sum]
s**13**sum=MAX[sum__s**13**, s**12**sum]
s**14**sum=MAX[sum__s**14**, s**13**sum]
s**15**sum=MAX[sum__s**15**, s**14**sum]
s**16**sum=MAX[sum__s**16**, s**15**sum]
s**17**sum=MAX[sum__s**17**, s**16**sum]

Control Logics—State Machine (CLSM) Module

As shown in FIG. **7**, the Control Logic module controls the overall operations of the Log-MAP Decoder. The control logic state machine **171**, referred as CLSM, is shown in FIG. **17**. The CLSM module **171** (FIG. **17**) operates effectively as follows. Initially, the CLSM module **171** operates in IDLE state **172**. When the decoder is enabled, the CLSM module **171** transitions to CALC-BM state **173**, where the Branch Metric (BM) module starts operations and monitors for completion. When Branch Metric calculations are completed, referred to as bm-done, the CLSM transitions to CALC-FWD-SM state **174**, where the State Metric module (SM) begins forward recursion operations. When the forward SM state metric calculations are completed, referred to as fwd-sm-done, the CLSM transitions to CALC-BWD-SM state **175**, where the State Metric module (SM) begins backward recursion operations. When backward SM state metric calculations are completed, referred to as bwd-sm-done, the CLSM transitions to CALC-Log-MAP state **176**, where the Log-MAP computation module begins calculating the maximum a posteriori (MAP) probability to produce soft decode output. When Log-MAP calculations are completed, referred to as log-map-done, the CLSM module **171** transitions back to IDLE state **172**.

BM Memory and SM Memory

The Branch-Metric Memory **74** and the State-Metric Memory **75** are shown in FIG. **7** as the data storage components for BM module **71** and SM module **72**. The Branch Metric Memory module is a dual-port RAM that contains M- bits of N memory locations as shown in FIG. **18**. The State Metric Memory module is a dual-port RAM that contains K-bits of N memory locations as shown in FIG. **19**. Data can be written into one port while reading at the other port.

Interleaver Memory

As shown in FIG. **4**, the Interleaver Memory A **43** stores data for the first decoder A **42** and Interleaver Memory B **45** stores data for the second decoder B **44**. In iterative pipelined decoding, the decoder A **42** reads data from Interleaver Memory B **45** and writes results data into Interleaver Memory B **43**, the decoder B **44** reads data from Interleaver Memory A **43** and write results into Interleaver Memory B **45**.

As shown in FIG. **20**, the De-Interleaver memory **41** includes a De-Interleaver module **201** and a dual-port RAM **202**, which contains M-bits of N memory locations. The Interleaver is a Turbo code internal interleaver as defined by 3 GPP standard ETSI TS 125 222 V3.2.1 (2000–05), or other source. The Interleaver permutes the address input port A for all write operations into dual-port RAM module. Reading data from output port B are done with normal address input.

As shown in FIG. **21**, the Interleaver Memory **43 45** comprises of a dual-port RAM **211**, which contains M-bits of N memory locations.

The input buffer Interleaver Memory module uses an interleaver to generate the writeaddress sequences of the Memory core in write-mode. In read-mode, the memory core readaddress are normal sequences.

Turbo Codes Decoder Control Logics—State Machine (TDCLSM

As shown in FIG. **4**, the Turbo Decoder Control Logics module **47**, referred to as TDCLSM, controls the overall operations of the Turbo Codes Decoder. Log-MAP A **42** starts the operations of data in Memory B **45**. At the same time, Log-MAP B starts the operations in Memory A **43**. When Log-MAP A **42** and Log-MAP B **44** finish with block N of data, the TDCLSM **47** starts the iterative decoding for L number of times. When the iterative decoding sequences are completed, the TDCLSM **47** transitions to HARD-DEC to generate the harddecode outputs. Then the TDCLSM **47** transitions to start decoding another block of data.

Iterative Decoding and Diversity Processing

Turbo Codes decoder performs iterative decoding and diversity processing by feeding back the output **Z1**, **Z3** of the second Log-MAP decoder B into the corresponding first Log-MAP decoder A before making decision for hard-decoding output. As shown in FIG. **23**, the Counter **233** counts the preset number L times.

We claim:

1. A baseband processing system for iteratively decoding data received on multiple data paths, the baseband processing system comprising:

at least one decoder adapted to receive the data received on one or more of the multiple data paths, wherein each decoder comprises:

at least two soft decision decoders adapted to receive data associated with corresponding data paths, wherein the at least two soft decision decoders are serially coupled and have at least a first soft decision decoder and a last soft decision decoder, wherein the last soft decision decoder is adapted to output data for the serially coupled series of soft decision decoders;

at least one memory module that is electrically coupled to an output of a corresponding soft decision decoder, wherein the output of the memory module associated with the last soft decision decoder is fed back as an input to the first soft decision decoder of each of the at least one decoders.

2. The system according to claim **1**, further comprising:

a control logic state machine that is adapted to control an operation of the at least one decoder when data is received at corresponding ones of the multiple data paths.

3. The system according to claims **2**, further comprising:

an adder that sums an output from each of the at least one decoders; and

a hard decoder that receives an output from the adder and provides a hard decoded output for the baseband processing system.

4. The system according to claim **1**, wherein the at least one decoder is a Turbo Code Decoder.

5. The system according to claim **1**, further comprising at least two decoders configured in a parallel arrangement.

6. The system according to claim **1**, wherein the soft decision decoder uses a logarithm maximum a posteriori probability algorithm.

7. The system according to claim **1**, wherein the soft decision decoder uses a Soft-input Soft-output method maximum a posteriori probability algorithm.

8. The system according to claim **1**, wherein the soft decision decoder uses a logarithm approximation algorithm.

9. The system according to claim **1**, wherein the at least one memory module comprises a dual-port random access memory.

10. The system according to claim **1**, wherein the at least one memory module comprises an interleaver memory having an interleaver that generates a write address sequence for a memory core in a write mode.

11. The system according to claim **1**, wherein the at least one memory module comprises an interleaver memory having an interleaver that generates a memory core having normal sequences in a read mode.

12. A method of iteratively decoding data received on multiple data paths using at least one decoder adapted to receive the data received on one or more of the multiple data paths, wherein each decoder comprises at least two soft decision decoders adapted to receive data associated with corresponding data paths, wherein the at least two soft decision decoders are serially coupled and have at least a first soft decision decoder and a last soft decision decoder, wherein the last soft decision decoder is adapted to output data for the serially coupled series of soft decision decoders, the method comprising:

receiving first soft decision data at a first decoder;

receiving second soft decision data at a second decoder;

utilizing a sliding window having a predetermined block size to process data received at the first decoder and data received at the second decoder;

providing the corresponding data processed by the sliding window at the first decoder to the associated at least two serially coupled soft decision decoders;

providing the corresponding data processed by the sliding window at the second decoder to the associated at least two serially coupled soft decision decoders;

performing, for a predetermined number of times, iterative decoding at the first and second decoders, wherein an output from the last soft decision decoder is fed back as an input to the first soft decision decoder of each of the first and second decoders.

13. The method according to claim **12**, further comprising controlling an operation of the first and second decoders when data is received at corresponding ones of the multiple data paths.

14. The method according to claim **13**, further comprising:

adding an output from each of the first and second decoders; and

providing hard decoded output data from the added data after performing the iterative decoding for the predetermined number of times.

15. The method according to claim **12**, wherein the at least two serially coupled soft decision decoders associated with the first and second decoders perform processing using a logarithm maximum a posteriori probability algorithm.

16. The method according to claim **12**, wherein the at least two serially coupled soft decision decoders associated with the first and second decoders perform processing using a Soft-input Soft-output method maximum a posteriori probability algorithm.

17. The method according to claim **12**, wherein the at least two serially coupled soft decision decoders associated with the first and second decoders perform processing using a logarithm approximation algorithm.

18. The method according to claim **12**, wherein utilizing the sliding window having the predetermined block size enables the data to be processed in a pipeline scheme.

19. The method according to claim **12**, wherein utilizing the sliding window having the predetermined block size enables the data to be processed in a continuous wraparound scheme for pipeline operations.

20. The method according to claim **12**, wherein the output of the last soft decision decoder is fed back to the first soft decision decoder for each of the first and second decoders in a pipeline mode to produce soft decoded data at each clock cycle.

21. The method according to claim **12**, wherein the at least two serially coupled soft decision decoders associated with each of the first and second decoders execute operations at a same time in a pipelined mode with a predefined delay latency.

22. A soft decision decoder comprising:

a branch metric module that is adapted to receive soft input data and is configured to compute branch metric values for each branch in a Trellis;

a branch metric memory module that is coupled to the branch metric module and is adapted to store data associated at least with the branch metric values;

a state metric module that is coupled to the branch metric memory module and is configured to compute state metric values for each state in the Trellis using the computed branch metric values;

an add-compare-select circuit that is coupled to the state metric module and is configured to compute state metric values at each node in the Trellis;

a state metric memory module that is coupled to the state metric module and is adapted to store data associated at least with the state metric values;

a computation module that is coupled to at least the branch metric memory module and the state metric memory module, wherein the computation module is configured to compute a soft decision output based at least on the branch metric values and the state metric values; and

a control logic state machine module that is adapted to control operations of at least one of the branch metric module, the branch metric memory module, the state metric module, the add-compare-select circuit, the state metric memory module, and the computation module.

23. The soft decision decoder according to claim **22**, wherein the computation module is implemented with a logarithm maximum a posteriori probability algorithm.

24. The soft decision decoder according to claim **22**, wherein the computation module is implemented with a Soft-input Soft-output method maximum a posteriori probability algorithm.

25. The soft decision decoder according to claim **22**, wherein the computation module is implemented with a logarithm approximation algorithm.

26. The soft decision decoder according to claim **22**, wherein the state metric module computes state metric values based on forward recursion.

27. The soft decision decoder according to claim **22**, wherein the state metric module computes state metric values based on backward recursion.

28. The soft decision decoder according to claim **22**, wherein the an add-compare-select circuit comprises:

a first adder for computing the sum of a first state metric value and a first branch metric value;

a second adder for computing the sum of a second state metric value and a second branch metric value;

a comparator for comparing the results of the first adder and the results of the second adder; and

a multiplexer for selecting a larger sum for a predetermined state.

29. A soft decision decoder comprising:

branch metric means for receiving soft input data and computing branch metric values for each branch in a Trellis;

branch metric memory means for storing data associated at least with the branch metric values;

state metric means for computing state metric values for each state in the Trellis using the computed branch metric values;

add-compare-select means for computing state metric values at each node in the Trellis;

state metric memory means for storing data associated at least with the state metric values;

computation means for computing a soft decision output based at least on the branch metric values and the state metric values; and

control logic state machine means for controlling operations of at least one of the branch metric means, the branch metric memory means, the state metric means, the add-compare-select means, the state metric memory means, and the computation means.

30. A method of implementing a soft decision decoder for decoding a plurality of data sequences, comprising:

receiving soft decision data;

utilizing a sliding window of a predetermined block size to process the soft decision data;

computing a branch metric for each data element of the soft decision data associated with the predetermined block size, wherein the branch metric is computed for branches entering each state in the Trellis;

computing a forward recursion state metric for each data element of the soft decision data associated with the predetermined block size, wherein the state metric is computed for each state in the Trellis;

computing a backward recursion state metric for each data element of the son decision data associated with the predetermined block size, wherein the state metric is computed for each state in the Trellis;

computing logarithm maximum a posteriori probability values based on at least the branch metric, the forward recursion state metric, and the backward recursion state metric for each data element of the soft decision data associated with the predetermined block size; and

providing soft-decisions based on the logarithm maximum a posteriori probability values.

31. The method according to claim **30**, wherein computing the branch metric for each data element comprises calculating a Euclidean distance for each branch.

* * * * *

US006799295C1

(12) **EX PARTE REEXAMINATION CERTIFICATE** (6707th)

United States Patent

Nguyen

(10) **Number: US 6,799,295 C1**

(45) **Certificate Issued: Mar. 17, 2009**

(54) **HIGH SPEED TURBO CODES DECODER FOR 3G USING PIPELINED SISO LOG-MAP DECODERS ARCHITECTURE**

(75) Inventor: **Quang Nguyen**, Allentown, PA (US)

(73) Assignee: **Icomm Technologies, Inc.**, Fogelsville, PA (US)

Reexamination Request:
No. 90/008,190, Aug. 25, 2006

Reexamination Certificate for:

Patent No.:	**6,799,295**
Issued:	**Sep. 28, 2004**
Appl. No.:	**10/248,245**
Filed:	**Dec. 30, 2002**

Related U.S. Application Data

(63) Continuation-in-part of application No. 10/065,408, filed on Oct. 15, 2002, now abandoned, which is a continuation-in-part of application No. 09/681,093, filed on Jan. 2, 2001, now Pat. No. 6,813,742.

(51) **Int. Cl.**

H03M 13/00	(2006.01)
H03M 13/29	(2006.01)
H04L 1/00	(2006.01)

(52) **U.S. Cl.** **714/794**; 714/755

(58) **Field of Classification Search** 375/240.25, 375/265, 341, 346, 354; 714/701, 746, 755, 714/780, 786, 792, 795, 807
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

6,023,783	A	*	2/2000	Divsalar et al.	714/792
6,141,384	A	*	10/2000	Wittig et al.	375/240.25
6,216,249	B1	*	4/2001	Bliss et al.	714/792
6,335,954	B1	*	1/2002	Bottomley et al.	375/354
6,351,832	B1	*	2/2002	Wei	714/701
6,360,345	B1	*	3/2002	Kim et al.	714/746
6,460,162	B1	*	10/2002	Buda et al.	714/807
6,487,694	B1	*	11/2002	Bajwa	714/786
6,499,128	B1	*	12/2002	Gerlach et al.	714/755
6,526,538	B1	*	2/2003	Hewitt	714/780
6,563,877	B1	*	5/2003	Abbaszadeh	375/242

(Continued)

FOREIGN PATENT DOCUMENTS

JP	08-032633	2/1996

OTHER PUBLICATIONS

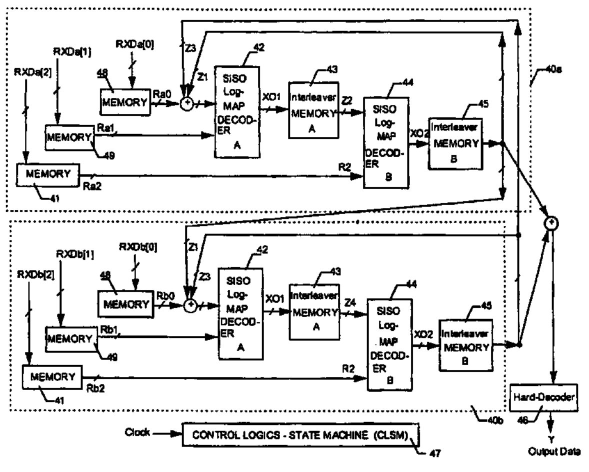
S. Crozier, K. Gracie, and A. Hunt, "Efficient Turbo Decoding Techniques," Proceedings of the 11th International Conference on Wireless Communications, Calgary, Alberta, Canada, pp. 187–195, Jul. 12–14, 1999.

(Continued)

Primary Examiner—Christopher E Lee

(57) **ABSTRACT**

A baseband processor is provided having Turbo Codes Decoders with Diversity processing for computing signals from separate antennas. The invention decodes multipath signals that have arrived at the terminal via different routes after being reflected from buildings, trees or hills. The Turbo Codes Decoder with Diversity processing increases the signal to noise ratio (SNR) more than 6 dB which enables the 3rd Generation Wireless system to deliver data rates from up to 2 Mbit/s. The invention provides several improved Turbo Codes Decoder methods and devices that provide a more suitable, practical and simpler method for implementation a Turbo Codes Decoder in ASIC or DSP codes. A plurality of parallel Turbo Codes Decoder blocks are provided to compute soft-decoded data RXDa, RXDb from two different receiver path. Several pipelined Log-MAP decoders are used for iterative decoding of received data. A Sliding Window of Block N data is used on the inputted data for pipeline operations. In a pipeline mode, a first decoder A decodes block N data from a first source, while a second decoder B decodes block N data from a second source during the same clock cycle. Pipelined Log-MAP decoders provide high speed data throughput and one output per clock cycle.



TURBO CODES DECODER SYSTEM BLOCK DIAGRAM

U.S. PATENT DOCUMENTS

6,598,204	B1	*	7/2003	Giese et al.	714/795
6,633,615	B1	*	10/2003	Pekarich et al.	375/265
6,671,338	B1	*	12/2003	Gamal et al.	375/346
6,690,737	B1	*	2/2004	Kim	375/265
6,732,327	B1	*	5/2004	Heinila	714/792
6,788,750	B1	*	9/2004	Reuven et al.	375/341
6,829,313	B1		12/2004	Xu	

OTHER PUBLICATIONS

T. Ngo and I. Verbauwhede, "Fixed Point Implementation for Turbo Codes," Final Report for 1998–99 Micro Project 98–162, http://www.ucop.edu/research/micro/98__99/98__162.pdf.

S. Pietrobon, "Implementation and Performance of a Turbo/Map Decoder," Int. J. Satellite Commun., vol. 16, pp. 23–46, Jan.–Feb. 1998.

J. Nikolic–Popovic, "Real–Time Implementation of cdma2000 Turbo Decoder on TMS320C6200," International Conferences on Signal Processing Applications and Technology, 1999.

L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate," IEEE Trans. Inform. Theory, vol. IT–20, pp. 284–287, Mar. 1974.

K. Tzou and J. Dunham, "Sliding Block Decoding of Convolutional Codes," IEEE Trans. Inform. Theory, vol. Com–29, pp. 1401–1403, Sep. 1981.

* cited by examiner

EX PARTE REEXAMINATION CERTIFICATE ISSUED UNDER 35 U.S.C. 307

THE PATENT IS HEREBY AMENDED AS INDICATED BELOW.

Matter enclosed in heavy brackets [ ] appeared in the patent, but has been deleted and is no longer a part of the patent; matter printed in italics indicates additions made to the patent.

AS A RESULT OF REEXAMINATION, IT HAS BEEN DETERMINED THAT:

Claims **30** and **31** are cancelled.

Claims **22** and **29** are determined to be patentable as amended.

Claims **23**–**28**, dependent on an amended claim, are determined to be patentable.

New claims **32**–**41** are added and determined to be patentable.

Claims **1**–**21** were not reexamined.

22. **[**A soft decision decoder comprising**]** *The system according to claim 1, wherein each of the at least two soft decision decoders comprises*:

a branch metric module that is adapted to receive soft input data and is configured to compute branch metric values for each branch in a Trellis *by calculating an Euclidean distance for each branch*;

a branch metric memory module that is coupled to the branch metric module and is adapted to store data associated at least with the branch metric values;

a state metric module that is coupled to the branch metric memory module and is configured to compute state metric values for each state in the Trellis using the computed branch metric values;

an add-compare-select circuit that is coupled to the state metric module and is configured to compute state metric values at each node in the Trellis;

a state metric memory module that is coupled to the state metric module and is adapted to store data associated at least with the state metric values;

a computation module that is coupled to at least the branch metric memory module and the state metric memory module, wherein the computation module is configured to compute a soft decision output based at least on the branch metric values and the state metric values; and

a control logic state machine module that is adapted to control operations of at least one of the branch metric module, the branch metric memory module, the state metric module, the add-compare-select circuit, the state metric memory module, and the computation module.

29. **[**A soft decision decoder comprising**]** *The system according to claim 1, wherein each of the at least two soft decision decoders comprises*:

branch metric means for receiving soft input data and computing branch metric values for each branch in a Trellis *by calculating an Euclidean distance for each branch*;

branch metric memory means for storing data associated at least with the branch metric values;

state metric means for computing state metric values for each state in the Trellis using the computed branch metric values;

add-compare-select means for computing state metric values at each node in the Trellis;

state metric memory means for storing data associated at least with the state metric values;

computation means for computing a soft decision output based at least on the branch metric values and the state metric values; and

control logic state machine means for controlling operations of at least one of the branch metric means, the branch metric memory means, the state metric means, the add-compare-select means, the state metric memory means, and the computation means.

32. The baseband processing system according to claim 1, wherein the output of the memory module associated with the first soft decision decoder is fed back as an input to the last soft decision decoder, and

wherein the at least one decoder further comprises:

at least one interleaver coupled to the memory module associated with the first soft decision decoder, and

at least one de-interleaver coupled to the memory module associated with the last soft decision decoder.

33. The baseband processing system according to claim 1, wherein the at least two soft decision decoders have at least the first soft decision decoder, the last soft decision decoder, and a second soft decision decoder serially coupled between the first soft decision decoder and the last soft decision decoder,

wherein the output of the memory module associated with the last soft decision decoder is fed back as an input to the first soft decision decoder,

wherein the output of the memory module associated with the first soft decision decoder is fed back as an input to the second soft decision decoder, and

wherein the at least one decoder further comprises:

at least one interleaver coupled to the memory module associated with the first soft decision decoder, and

at least one de-interleaver coupled to the memory module associated with the second soft decision decoder.

34. The baseband processing system according to claim 1, wherein the data is received on the multiple data paths from one or more antennas and pre-processed by one or more receivers, and

wherein at least one of the at least two soft decision decoders is adapted to iteratively process the pre-processed data.

35. A baseband processing system according to claim 1, wherein the data is received on the multiple data paths from a single antenna,

wherein the output of the memory module associated with the first soft decision decoder is fed back as an input to the last soft decision decoder, and

wherein the at least one decoder further comprises:

at least one interleaver coupled to the memory module associated with the first soft decision decoder, and

at least one de-interleaver coupled to the memory module associated with the last soft decision decoder.

36. The method according claim 12, wherein performing, for a predetermined number of times, iterative decoding at the first and second decoders comprises:

processing the data received on the multiple data paths using an algorithm comprised of a log-MAP algorithm or a logarithm approximation algorithm to generate a soft decision;

storing the soft decision;

performing iterative decoding from the first soft decision decoder to the last soft decision decoder in circular circuit for the predetermined number of times;

determining a hard decision output.

37. A method of processing according claim 12, wherein the data is received on the multiple data paths from a plurality of antennas,

wherein performing, for a predetermined number of times, iterative decoding at the first and second decoders comprises:

processing the data received on the multiple data paths using an algorithm comprised of a log-MAP algorithm or a logarithm approximation algorithm to generate a soft decision;

storing the soft decision;

performing iterative decoding from the first soft decision decoder to the last soft decision decoder in circular circuit for the predetermined number of times;

determining a hard decision output.

38. The method according to claim 12, wherein the data is received on the multiple data paths from a plurality of antennas,

wherein utilizing a sliding window having a predetermined block size to process data received at the first decoder comprises utilizing a sliding window having a predetermined block size for pipelined processing of soft decision data from the first soft decision decoder to the second soft decision decoder iteratively.

39. The method of claim 38, wherein utilizing a sliding window having a predetermined block size for pipelined processing of soft decision data from the first soft decision decoder to the second soft decision decoder iteratively comprises:

computing a branch metric for each data element of the soft decision data in the sliding window for branches entering each state in the trellis by determining a Euclidean distance for each branch starting from the first state of the trellis to the last state of the trellis:

computing state metrics in a forward direction starting from the first state of the trellis to the last state of the trellis, the computing state metrics in a forward direction comprising:

calculating a sum of each branch metric plus a state metric entering a corresponding state of the trellis, and

selecting a maximum value between two entering state metrics corresponding to a first path and a second path;

computing state metrics in backward direction starting from the last state of the trellis to the first state of the trellis, the computing state metrics in a backward direction comprising:

calculating a sum of each branch metrics plus a state metric entering a corresponding state of the trellis, and

selecting a maximum value between two entering state metrics corresponding to a first path and a second path; and

computing the soft decision output for each state in the trellis, starting from the first state of the trellis to the last state of the trellis; and

storing the soft decision output for the performing iterative decoding.

40. A soft decision decoder comprising:

a branch metric module that is adapted to receive soft input data and is configured to compute branch metric values for each branch in a Trellis, wherein the soft input data includes a first soft input and a second soft input, wherein the branch metric values consist of a first set of branch metric values, a second set of branch metric values, a third set of branch metric values, and a fourth set of branch metric values, wherein the first set of branch metric values are determined as a sum of the first soft input and the second soft input, wherein the second set of branch metric values are determined as a difference between the first soft input and the second soft input, wherein the third set of branch metric values are determined as a twos compliment of the sum of the first soft input and the second soft input, wherein the fourth set of branch metric values are determined as a twos compliment of the difference between the first soft input and the second soft input;

a branch metric memory module that is coupled to the branch metric module and is adapted to store data associated at least with the branch metric values;

a state metric module that is coupled to the branch metric memory module and is configured to compute state metric values for each state in the Trellis using the computed branch metric values;

an add-compare-select circuit that is coupled to the state metric module and is configured to compute state metric values at each node in the Trellis;

a state metric memory module that is coupled to the state metric module and is adapted to store data associated at least with the state metric values;

a computation module that is coupled to at least the branch metric memory module and the state metric memory module, wherein the computation module is configured to compute a soft decision output based at least on the branch metric values and the state metric values; and

a control logic state machine module that is adapted to control operations of at least one of the branch metric module, the branch metric memory module, the state metric module, the add-compare-select circuit, the state metric memory module, and the computation module.

41. A soft decision decoder comprising:

a branch metric module that is adapted to received soft input data and is configured to compute branch metric values for each branch in a Trellis, wherein the soft input data includes a first soft input and a second soft input, wherein all the branch metric values are determined based on either a sum of the first soft input and the second soft input or a difference between the first soft input and the second soft input;

a branch metric memory module that is coupled to the branch metric module and is adapted to store data associated at least with the branch metric values;

a state metric module that is coupled to the branch metric memory module and is configured to compute state metric values for each state in the Trellis using the computed branch metric values;

an add-compare-select circuit that is coupled to the state metric module and is configured to compute state metric values at each node in the Trellis;

a state metric memory module that is coupled to the state metric module and is adapted to store data associated at least with the state metric values;

a computation module that is coupled to at least the branch metric memory module and the state metric memory module, wherein the computation module is configured to compute a soft decision output based at least on the branch metric values and the state metric values; and

a control logic state machine module that is adapted to control operations of at least one of the branch metric module, the branch metric memory module, the state metric module, the add-compare-select circuit, the state metric memory module, and the computation module.

* * * * *